

Europäisches Patentamt
European Patent Office
Office européen des brevets



(11) EP 0 740 450 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
30.10.1996 Bulletin 1996/44

(51) Int Cl.⁶: H04M 3/50, H04Q 3/66

(21) Application number: 96302807.1

(22) Date of filing: 22.04.1996

(84) Designated Contracting States:
AT CH DE FR GB IT LI

(30) Priority: 24.04.1995 US 427546

(71) Applicants:

- INTERNATIONAL BUSINESS MACHINES CORPORATION
Armonk, NY 10504 (US)
- SIEMENS ROLM COMMUNICATIONS INC.
Santa Clara, CA 95054 (US)

(72) Inventors:

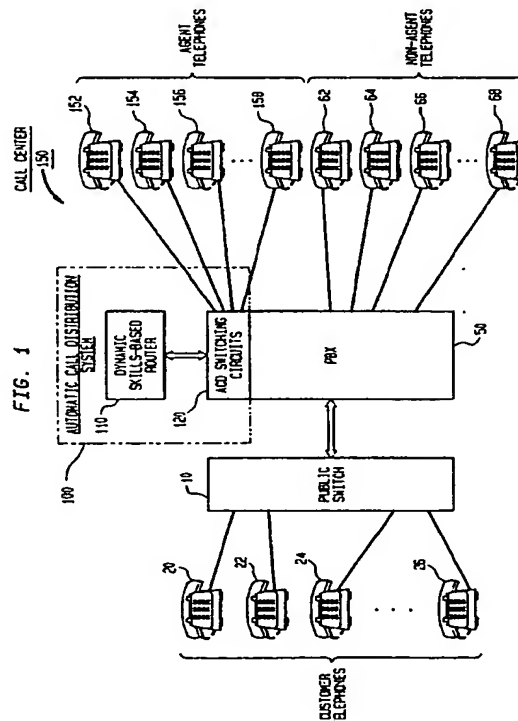
- Brooks, Nancy S.
Markham, Ontario, L3R 3H6 (CA)

- Fryer, Paul D.
Toronto, Ontario, M4A 1L8 (CA)
- Kaufman, Gary S.
Thornhill, Ontario, L4J 8A6 (CA)
- Berkson, Stephen P.
Los Gatos, California 95030 (US)
- Herel, Charles R.
Pleasanton, California 94588 (US)
- Brooks, Laura M.
Sunnyvale, California 94086 (US)

(74) Representative: Williams, Julian David
IBM United Kingdom Limited,
Intellectual Property Department,
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Method and apparatus for skill-based routing in a call center

(57) An automatic call distribution (ACD) system has a transaction queue for storing data indicating the skills and the levels of these skills that are required to process the calls at a call center. The ACD system also has a skills inventory database for storing data indicating the skills and the skill levels of each of the agents at the call center. The skills inventory database also stores data indicating the preferences and the preference levels of the call center management to have specific agents process calls with specific skill requirement levels. The ACD system compares the skill levels required by a call with the skill levels and/or preference levels of available agents and distributes the call to the available agent that has the best skill and/or preference match. A match is best with respect to skills if the agent is not underqualified to process the call and if the agent is the least overqualified agent. A match is best with respect to preferences if the call center management most prefers that the agent handle the call. The call center management controls whether the ACD system considers skill matches, preference matches, or both.



EP 0 740 450 A2

Description**Technical Field of the Invention**

5 The present invention relates to the routing of telephone calls to agents at a call center. More specifically, the present invention relates to the routing of telephone calls based on the respective skills of the agents and the skill requirements of the telephone calls.

Background of the Invention

10 The present invention relates to a method and apparatus for distributing work between telephone agents (agents) in a call center (center) of a business enterprise (company). A call center is an organization set up by a company to handle large volumes of telephone-based business transactions. Call centers are currently commonly used by companies world-wide in order to provide an efficient alternative to performing business transactions with customers in a face-to-face manner. For example, rather than renting one or more commercial spaces in shopping areas in order to retail merchandise to the general public, a company may mail out catalogs which advertise a telephone number that can be used for ordering the same merchandise. Customers use the catalog to select the merchandise they are interested in, and then dial the advertised number. If the retailer is successful in generating demand for its products, then the retailer will receive a large number of telephone calls. Without any specific organization to handle these telephone calls, the company's efforts may be wasted. In order to complete each sales transaction, a company employee must be available to speak with a caller, answer the caller's questions, solicit some form of payment, and launch an internal business process which will result in the fulfilment of the caller's request and the gathering of payment. By definition, an employee of a company is considered a telephone agent (agent) during the period of time in which the employee's assigned work is organized into business transactions which are conducted over the telephone. The telephone calls over which the agent conducts each transaction may be initiated either by the customer (in which case the transaction is referred to as an inbound transaction), or by the agent (in which case the transaction is referred to as an outbound transaction). Although the present invention relates to both inbound and outbound transactions, the description of the present invention focuses on inbound transactions. Also, the customer may be a member of the same company (as when one organizational sub-unit of a company provides service to another), or a non-member of the company (as one typically thinks of the term customer, as a person who may desire to purchase goods and/or services from the company).

In order to enable the handling of telephone calls, call centers invest in telecommunications equipment which is capable of making voice connections between customers and agents. Sometimes this equipment is physically located on the company's premises, while in other cases the company may have made arrangements with a telephone service provider (for example, a Regional Bell Operating Company) to use equipment which is located completely or partially on telephone service provider premises. The preferred embodiment of the present invention is implemented on equipment which resides on a company's premises, and which shall be referred to as the switch. However, the present invention can also be implemented in a similar manner on equipment resident at telephone service provider locations.

40 Having so defined call centers, and the employees that work in them as agents, and the equipment that agents use as telephones attached to switches, we can now consider some of the problems encountered in the management of telephone-based transactions.

First of all there is the natural requirement to provide some method for distributing the workload. Consider a center with a number of agents. Each time an inbound transaction arrives at the center (i.e. a call arrives), we must have some strategy for deciding which of the agents should be assigned to handle this new transaction. In the past, a primitive approach was to set up a sequencing scheme (either explicitly using commands to the switch, or implicitly just by how the telephones were wired) so that the switch selected which agent would handle the call by checking in sequence, whether or not each agent was available to take an inbound call (i.e. if a first agent is not busy, the first agent takes the call. If the first agent is busy, then if a second agent is not busy, then the second agent takes the call. If the second agent is busy, then if a third agent is not busy and so on). Using such a sequencing scheme led to the problem of creating hot-seats; where those agents at the beginning of the order tended to do more work (i.e. take more calls) than agents toward the end of the sequence.

55 In order to avoid the problem of hot-seats (i.e. inequitable distribution of the inbound transaction load amongst the agents) most switches now select which agent will receive the next inbound transaction by using a longest idle agent method. Using this scheme, the switch keeps track of exactly how long each agent has been idly waiting for their telephone to ring with an inbound transaction. When an inbound transaction arrives at the company, the switch sends it to the agent who has been idle the longest.

Note that the center is unable to control exactly when it receives inbound transactions, nor how many it receives at any instant in time. This is because each call results from a decision made by a separate human being, acting for

the most part independently of all other callers. As a result of this inherent unpredictability, we can postulate that there will exist a peaktime at which the call center experiences its maximum load of inbound transactions (which we will refer to as maxload). In order to ensure that a caller to the call center can always interact with an agent without procedural delay, a company could choose to employ enough agents so that the number of agents prepared to accept an inbound transaction at any instant in time, was never less than maxload. In practice however, this level of staffing often proves to be prohibitively expensive, as the postulated peaktime may only occur rarely (so that many employees are often idle), and its actual value varies along with business conditions, and is difficult to predict. Most companies therefore accept the fact that there will be times when a caller seeking to perform an inbound transaction, will call at a time when all of the agents are already engaged in inbound transactions with other callers. These companies then plan strategies to minimize the dissatisfaction of such customers whom they are unable to provide service to, without some sort of procedural delay. One common strategy is to allow customers to wait in a queue for an available telephone agent.

Using this approach, when all agents are busy servicing callers, new calls which arrive are answered, but not by an agent; instead either a pre-recorded message or music, or a live radio signal, or some combination of all of these, is played to the caller, while their call is held, pending the availability of an agent. As soon as an agent becomes available, the call which has been waiting longest in the queue is routed by the switch to the newly available agent. If two or more agents become available at exactly the same time, then the switch can select randomly which of them gets the call.

Up to this point we have assumed that any agent in the center is able to handle any business transaction. In practice, this is an oversimplification, as most companies organize their agents into work groups which are oriented towards handling specific types of business transactions. As a simple example, consider how a company can provide support of two different languages, such as English and Spanish. First the company finds out which of its agents speak each language. Then it divides the agents into three groups: English-speaking, Spanish-speaking, and bilingual. The company may then choose to advertise two different telephone numbers: one for service in English, and another for service in Spanish. When inbound transactions arrive, if they have been dialed on the English number, then they queue for an available English-speaking agent (i.e. an agent in the English group). If they have been dialed on the Spanish number, then they are queued by the switch for the Spanish group. The term group refers to a set of agents who are collectively considered by the switch as suitable for receiving an inbound transaction of a certain type or types, or placing an outbound transaction, of a certain type or types. Some switches can search for available agents in more than one queue, so in the example above, we could create a third group of bilingual agents. If all agents in the Spanish group are busy, then an inbound transaction might be sent to the bilingual agents group, as they are capable of performing business transactions using the Spanish language. If all agents in both of these groups are busy, then the switch can hold a place in queue for both groups, and wait for the next available agent, whether that agent is in the Spanish group or the bilingual group. Note that calls queuing for the bilingual group, may be for English language business transactions also.

For the purposes of this description, we shall refer to the switch components (collectively) which enable these functions for assembling agents into groups, directing inbound calls to one or more specific groups, and holding calls in queues for one or more groups, as an automatic call distribution system (or ACD). ACD systems typically include numerous features which allow for variations in how telephone-call-based transactions are handled, such as functions for varying which groups answer which calls, based on the time of day, or the day of the week.

The major purpose of any call center is to handle large volumes of telephone-based transactions in a cost-effective manner, while providing satisfactory customer service, and maintaining satisfactory agent morale. Current ACD systems contribute to this purpose by automating two major decisions for the company managing the call center:

- 1) The Agent Selection Decision: When more than one agent is available to handle the next transaction, which agent should be chosen?
- 2) The Transaction Selection Decision: When more than one call or transaction is waiting for the next available agent and an agent finally does become available, which call (or transaction) should the agent handle?

As explained above, most current ACD systems solve the agent selection decision by using a longest idle agent strategy.

The basis of most current solutions for the transaction selection decision is to create one or more first-in, first-out (FIFO) queues. Using this solution, each call is marked with a priority level by the switch. When an agent becomes available, the call with the highest priority is transferred to that agent. If several calls of equal priority are waiting, then the call which has been waiting the longest is transferred (i.e. amongst calls of the same priority level, the ordering is FIFO).

When the center is using ACD for performing outbound transactions, each transaction is similarly submitted to a FIFO queue with some kind of priority designation, and then the switch routes transactions from this queue to agents (and may or may not perform an outbound dialing operation at the same time, to reach the customer identified in the

outbound transaction). Thus, both the agent selection decision and the transaction selection decision must be made in some form or another in order to perform both inbound and outbound transactions.

If all telephone-based transactions handled by a call center were alike (e.g. if each transaction were a credit-card purchase of the exact same product), then the company managing the center could be easily assured that as long as each agent passed a mandatory training program (certifying that they were able to perform the single valid transaction), each agent would be able to handle any telephone-based transaction. In practice, however, this is rarely the case. Most companies must deal with a great variety of inbound transactions in the same call center. Consider some examples:

1) A Brokerage Call Center: Agents in the center are able to buy and sell stocks and bonds, sell company services, send out flyers and booklets and other information, and provide investment advice. The same agent might in one transaction help a customer to buy shares of a stock, and then in the next transaction help another customer to open a new account.

2) A Product Support Call Center: Agents in the center are able to create, work on, escalate, and close problem reports, all for a variety of different products, each requiring special training. The same agent might in one transaction open a problem report for a customer about a first product, and then, in the next transaction, help another customer to close a problem report on a second product.

3) A Utility Customer Service Center: Agents in the center are able to turn on or off service to customers, answer questions or make changes to customer bills, and dispatch repair crews. The same agent might in one transaction answer a question about a monthly utility bill, and then, in the next transaction, determine that a customer's complaint should result in a repair crew being dispatched.

Most current ACD systems make the decisions of agent selection and transaction selection as described above (using longest idle agent and FIFO methods, respectively). This approach works well when there is little variation in the types of transactions being handled by the agents, because, as long as any agent can handle any transaction, the center can be assured of approaching some optimal level of transaction-thruput (i.e. the number of transactions handled by the entire center in a particular time interval), simply by minimizing any procedural delays involved in connecting customers with agents. In other words, the ACD systems described above act to ensure that the center begins work on a transaction as soon as possible. They do not however, act to ensure that procedural delays are minimized after an agent begins to work on the transaction. In practice, as can be easily observed when one considers the examples above, not all agents are equally adept at performing all transactions. As a result, current ACD systems are unable to maximize the transaction thruput of the center as a whole, as they suffer from the following deficiencies:

1) Under-Skilled Agent Problem

2) Over-Skilled Agent Problem

The Under-Skilled Agent Problem

The switch may route transactions to agents who do not have sufficient skills to handle the transactions. For example, some agents may have been only recently hired, and so have not yet been completely trained. Also, certain transactions rarely occur so that not all agents have enough experience to handle these transactions. Also to be considered, are the cases when the management of the center decides that it is prohibitively expensive to train all agents to handle all transactions, so instead they train specialists (i.e. agents who have additional training, beyond that of other agents, for handling certain types of transactions).

In all of these cases, most current ACD systems will on occasion route transactions to agents who are unable to handle them. When this occurs, the agent must apologize to the customer requesting the transaction, and transfer the call to another agent who is capable of helping the customer. This procedural delay is often exacerbated as the customer may vent some anger upon the agent, further delaying the processing of the transaction. Also, the agent may have to transfer the customer to another ACD group, where the customer may again have to wait in an ACD queue. Further, the agent may have difficulty finding another agent to whom the call can be transferred. The agent may have to put the customer on hold and begin searching either by phone or on foot, moving from agent to agent, asking whether each agent is able to take the call. Finally, even after the customer has already provided some or all of the information about the transaction to the first agent; now either the customer or the first agent, must repeat this information to the second agent.

In many situations, substantial adverse consequences can result from directing a transaction to an agent lacking sufficient training to complete the transaction. In all cases, the first agent spends valuable time unnecessarily correcting the error, time which might have been spent completing one or more transactions of another kind. In addition, in many cases the customer is angered by the error, which may lead to more long term consequences including loss of revenue

if the customer decides not to continue doing business with the company.

Using the scenarios provided above, here are some examples of the under-skilled agent problem:

1) A Brokerage Call Center: A call from a customer interested in investing in commodities is directed to an agent who is only trained regarding the purchase of stocks, bonds, and mutual funds. The agent may even be currently enrolled in a several month long course about commodities trading, but has not yet successfully completed the course.

2) A Product Support Call Center: A call from a customer about a first product reaches an agent who is trained in the first product, but once they begin working on the problem, the agent determines that the problem is beyond the agent's capabilities. For example, the agent may have only been working on transactions related to the first product for a few months, and the problem may be very difficult. In this situation, the call must be transferred to an agent that has more experience with the first product.

The Over-Skilled agent Problem

The switch may route transactions to agents who possess skills far in excess of those required to handle the transactions satisfactorily. In these cases we say that the agents are over-skilled, meaning that they are over qualified, and implying that their training and experience is wasted upon the work delegated to them. The routing of transactions to over-skilled agents may occur because some agents in a center who have had vast amounts of on the job training, are included in groups with other, less-experienced agents. It may also occur when the call center management chooses to designate certain agents as specialists, i.e. these agents are selected to receive specialized training in handling certain types of transactions over and above the base-level training normally supplied to agents. From the perspective of the center's management, it may be too expensive to train all agents to be experts in all transactions, but it may be viable to create a number of specialist agents possessing more extensive expertise in certain areas.

In all of these cases, most current ACD systems will on occasion route transactions to agents who are over-qualified to handle them. When this occurs, the over-skilled agent receiving such a transaction still handles the transaction as normal, but, assuming that other calls, which require the greater-than-average skill level of this agent, are arriving at the center at the same time, consider the effects upon the center as a whole:

1) At a first time, a first transaction is routed to a first agent, who is over-skilled to handle the first transaction. Assume the first agent has been specially trained to handle more difficult transactions. Although the first agent is able to work on the first transaction, because it is a simple transaction, so were a number of other agents that were also available at the first time. However, the switch, using the longest idle agent method, routed the first transaction to the first agent because the first agent had been idle the longest.

2) At a second time, just a few seconds after the first time, a second transaction arrives, which is very difficult, and can only be handled by agent the first agent. However, the first agent is unavailable to accept the second transaction because the first agent is working on the first transaction. (Even if the first agent were available, the switch would only have routed the second transaction to the first agent if the first agent had been the longest idle agent at the time). So the switch routes the second transaction to a second agent, who is the longest idle agent at the second time. But the second agent is not sufficiently skilled to handle the second transaction, so, as a result, the center experiences the under-skilled agent problem, as described above.

Thus, under certain loading conditions, transactions routed to over-skilled agents result in the switch actually creating new instances of the under-skilled agent problem, with all of the serious consequences that this problem entails. In practice, these loading conditions occur commonly and are another force which acts to diminish the transaction thruput of the center.

Another, less obvious cost of the over-skilled agent problem, is the opportunity cost of the training provided to specialist agents. Often times such training represents a substantial financial investment by the center's management. If, after all the training costs have been incurred, specialist agents spend significant portions of their time working on transactions which could have been handled without the specialized training, then to some extent, the company is not receiving full value for their training investment. The company did not have to train as many people to be specialists.

As described above, current ACD systems allow agents to be grouped together for the purposes of making agent selection decisions for dispatching transactions. Using the examples provided above, we can consider some of the ways in which agents can be divided up into groups:

1) A Brokerage Call Center: As an example, agents in this center could be divided up into two groups, a stocks and bonds group and a commodities group.

2) A Product Support Call Center: As an example, agents in this center could be divided up into groups according

to products as follows:

Product A group
Product B group
Product C group
Product D group

3) A Utility Customer Service Center: As an example, agents in this center could be divided up into two groups, a credit and collections group and a sales and service group.

Dividing agents into groups helps a center to avoid, to some extent, the under-skilled agent problem described above. However, there is still likely to be some occurrences of the under-skilled agent problem, and this method also has several other drawbacks. For example, call center managers must spend substantial amounts of time determining an appropriate division of agents into groups. Call center management must first perform a planning exercise in order to estimate the configuration most likely to meet the company's objectives, and then manually enter numerous commands into the switch's control console in order to have the switch manifest this plan. Once this configuration has been stored in the switch's database, the switch will handle all transactions using this configuration. Typically, call center management must repeatedly monitor the performance of the call center and modify the configuration to improve performance. For example, the configuration must usually be modified when there is a change in the size, pace and mix of transaction loads.

Planning an optimal configuration of a total number of agents (N) into a number of groups is not usually a simple task. Consider example (2) above (A Product Support center). The example, (which will be called configuration I) seems simple enough as described thus far: the company trains each agent to be expert in handling transactions for a single product. Then the company need only divide the agents into groups, based on product. If the company has four products, as shown above, then the company has four groups. The sum of all of the agents in all groups remains equal to the total number of agents, N.

Let us work with this example to demonstrate how, in practice, the situation is rarely this simple. First, let us consider the fact that the company may wish to support customers who speak only Spanish. In order to satisfy this realistic requirement, the company would now have to divide its agents into groups as follows (which will be referred to as configuration II):

1) English-only product A group
2) English-only product B group
3) English-only product C group
4) English-only product D group
5) Bilingual product A group
6) Bilingual product B group
7) Bilingual product C group
8) Bilingual product D group

Suddenly the company has doubled the number of groups it will require in order to support its customers. The configuration tasks are more complex now also, as the call center management must configure the switch to designate that if all agents in group 1 (English-only product A) are busy, that a transaction for group 1 can instead be sent to an agent in group 5 (Bilingual product A), as all of the agents in group 5 are sufficiently skilled (i.e. they speak English, and they are trained on product A), and so on for the other groups. Note that the company must do this extra configuration or face hiring additional agents. To appreciate this dilemma, consider how agents are distributed into groups. In configuration I, they are distributed as follows:

$$G1=N-(G2+G3+G4)$$

$$G2=N-(G1+G3+G4)$$

$$G3=N-(G1+G2+G4)$$

$$G4=N-(G1+G2+G3)$$

where $N=G1+G2+G3+G4$ and
 N, G1, G2, G3, and G4 are by definition greater than zero.
 In configuration II, they are now distributed across a larger number of groups, as follows:

$$G1=N-(G2+G3+G4+G5+G6+G7+G8)$$

$$G2=N-(G1+G3+G4+G5+G6+G7+G8)$$

$$G3=N-(G1+G2+G4+G5+G6+G7+G8)$$

$$G4=N-(G1+G2+G3+G5+G6+G7+G8)$$

$$G5=N-(G1+G2+G3+G4+G6+G7+G8)$$

$$G6=N-(G1+G2+G3+G4+G5+G7+G8)$$

$$G7=N-(G1+G2+G3+G4+G5+G6+G8)$$

$$G8=N-(G1+G2+G3+G4+G5+G6+G7)$$

where $N=G1+G2+G3+G4+G5+G6+G7+G8$ and
 N, G1, G2, G3, G4, G5, G6, G7, and G8 are by definition greater than zero.

In order to avoid hiring additional staff, the company keeps the total number of agents in the center at the level N, so that, if the company does not do additional configuration work (i.e. does not set up a scheme for groups to back each other up during busy times), then if there is suddenly a greater than normal number of transactions for a particular group (e.g. group 1), there will be less agents available to handle these transactions in configuration II, than in configuration I, and callers will have to wait longer in queue, and may potentially get frustrated and hang up (i.e. G1 for configuration II, is generally smaller than G1 for configuration I).

Now let us add another realistic factor to this scenario. Consider the situation that arises when the company comes out with a new product, product E. The company may choose to hire some additional agents, but they may also train some agents from other groups to be able to handle transactions for product E. A reasonable decision would be for the company to decide that all bilingual agents in the product D group, will now also be trained to work on transactions pertaining to product E, so as to minimize the hiring required for the product E group, until such time as the company can be sure that product E will be a success. The resulting configuration (configuration III) would then be:

- 1) English-only product A group
- 2) English-only product B group
- 3) English-only product C group
- 4) English-only product D group
- 5) English-only product E group
- 6) Bilingual product A group
- 7) Bilingual product B group
- 8) Bilingual product C group
- 9) Bilingual product D & E group

Once again human intervention is required to configure a new group, and to set up reasonable backup schemes for heavy load periods (e.g. group 9 can backup group 5 in configuration III above).

Finally, let us consider the most realistic of these scenarios, one where each agent is trained on one or more products. Products are often inter-related, and agents need to understand several in order to do a good job of solving technical problems for any one of them. When a new agent joins the center, they may initially be only trained on one product, but over a period of months and years, they gradually learn several others. Configuration IV (below) illustrates an example of such a center:

- 1) English-only product A only group
- 2) English-only product B only group
- 3) English-only product C only group
- 4) English-only product D only group
- 5) English-only product E only group
- 6) English-only products A & B group
- 7) English-only products C & D group
- 8) English-only experts on all products group
- 9) Bilingual product A only group
- 10) Bilingual product B only group
- 11) Bilingual product C only group
- 12) Bilingual product D only group
- 13) Bilingual product E only group
- 14) Bilingual products A & B group
- 15) Bilingual products C & D group
- 16) Bilingual experts on all products group

Even this configuration is over-simplified, as many other permutations can be imagined to provide a configuration in support of a product support call center supporting five products and two languages. Indeed, in practice such centers usually support tens of products. But this configuration does serve the overall purpose of these examples, which is to demonstrate that prior art grouping methods lead to labor-intensive approaches to organizing the work load of the agents in the call center, which requires constant maintenance of the configurations used for routing transactions to agents. Even with constant maintenance, call center management can never be sure that the optimal configuration has been used, as the transaction load for which the configuration was designed may be changing constantly.

Although skills-based grouping of agents in a call center may reduce the frequency of occurrence of the under-skilled agent problem, this method does not eliminate the problem altogether, and the method also has several other negative effects upon the center. For example, call center management is forced to perform a planning exercise to devise an acceptable configuration of agents into groups. This may be a significant effort requiring several individuals to work over a period of several days. Some companies will invest in hiring consultants in order to help perform this exercise. Thus, this planning activity costs the company substantial amounts of money. Also, the configuration decided upon may not be optimal for all situations. As the size, pace, and mix, of transactions change, there is a greater likelihood that the effects of the under-skilled agent and over-skilled agent problems will impact the performance of the center, as described above. Finally, in order to respond to changes in the size, pace, and mix of the transaction load, call center management must, on an ongoing basis, monitor the performance of the current grouping configuration, and manually make changes as necessary. Most companies find that the call center managers and supervisors must spend time each day monitoring and adjusting how many agents, and which agents, have been assigned to each group. Again, these activities consume valuable company resources that could otherwise be used to the competitive advantage of the company.

The "Quality of Service" Measurement Problem

In traditional ACD environments, Call Center managers manage their call center and evaluate the level of service provided to their clients based on several key indicators:

- Average Speed of Answer - The average time callers waited in a queue to be answered.
- Service Level Percentage - The percentage of calls answered within a defined service level.
- Abandon Percentage - The percentage of the call received that were abandoned.
- Average Time to Abandon - The average time callers waited in a queue before abandoning.

The call center manager is faced with the challenge of determining the adequate level of staffing required for each static group based on projected size, pace, and mix of the transaction load to meet their service level commitments to their customers and to minimize the percentage of calls that abandon.

For example, a health insurance company has contractual commitments with client A that 90% of their medical insurance claims calls will be answered with in a service level of 20 seconds and that 95% of their life insurance calls will be answered with in a service level of 30 seconds. The insurance company has similar contractual commitments to client B to handle 95% of its medical insurance calls in 20 seconds. For both clients the insurance company has made a commitment that less than 3% of calls received will be abandoned.

In the above scenario, the call center manager expends a great deal of effort to plan and staff to provide the appropriate level of service to each of their clients for each of their call types. In reality, the call center manager is really only addressing one aspect of the level of service provided to their customers. The traditional four service level indicators listed above address the speed of service and the clients tolerance to wait for service not the quality of service provided. As discussed in the "Under-Skilled Agent" Problem section above, call centers may route the call to an agent that does not have sufficient skills to service the client to meet service level goals. The under-skilled agent then either provides a low quality of service, places the caller on hold and consults with an agent with the appropriate skills, or ends up transferring the client to an agent with the appropriate skills. Traditional service level indicators would reflect that a high level of service was provided (95% of calls were answered in 20 seconds) when in reality customer satisfaction was very low.

The invention described in this application addresses this "Quality of Service" problem by providing call center managers with a quantitative measure of the quality of service provided to their clients for specific transaction types.

The "Agent Utilization" Measurement Problem

In traditional ACD environments, call center managers evaluate how effectively they are utilizing their agent resources by the percentage of the agent's available time that is spent handling ACD calls. As with the "Quality of Service" problem discussed above this indicator only covers one aspect of the agents utilization, the time factor. It does not address the skill factor. For example, the following questions can not be answered by traditional ACD agent utilization indicators:

How effectively did I utilize my agents skills?

Did they handle calls for which they were underskilled or over-skilled ?

Are they handling calls the type of calls which will increase their skill level?

How is handling efficiency affected when calls are handled by agents with different skill levels?

The invention described in this application addresses these aspects of the "agent utilization" problem by providing call center managers a quantitative measure of how closely the callers requirements were met by the agent who handled the call.

Summary of the Invention

The present invention relates to an automatic call distribution system for distributing a transaction between a plurality of agents at a call center. The automatic call distribution system comprises a transaction queue, a skills inventory database, a transaction dispatcher and a switching circuit. The transaction queue stores a set of skill requirement data for the transaction. The skills inventory database stores a set of skill data for each of the plurality of agents. The transaction dispatcher compares the set of skill requirement data for the transaction with the skill data for the agents and selects one of the agents for processing the transaction. The switching circuit is responsive to the transaction dispatcher for switching the transaction to the selected agent.

Brief Description of the Figures

Figure 1 is a functional block diagram of a telecommunication system that implements the preferred embodiment of the present invention.

Figure 2 is a functional block diagram of a call center and an automatic call distribution system implementing a preferred embodiment of the present invention.

Figure 3 is a flow chart illustrating a method performed by a transaction dispatcher of the preferred embodiment of the present invention.

Figures 4A and 4B are flow charts illustrating a pair of skill-only methods for matching an agent with a transaction of the preferred embodiment of the present invention.

Figure 5 is a flow chart illustrating a skill/preference method for matching an agent with a transaction of the preferred embodiment of the present invention.

Figure 6 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating

a skill score for a skill expression without parentheses.

Figure 7 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a skill score for a skill expression containing AND operators and OR operators.

Figure 8 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a skill score for a specific agent and a specific skill.

Figure 9 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a preference score for a skill expression without parentheses.

Figure 10 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a preference score for a skill expression containing AND operators and OR operators.

Figure 11 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a preference score for a specific agent and a specific skill.

Detailed Description

Figure 1 is a functional block diagram of a telecommunication system that implements the preferred embodiment of the present invention. The telecommunication system facilitates inbound telephone calls or transactions from customers to agents of a company's call center as well as outbound telephone calls or transactions from call center agents to customers. A number of customer telephones 20, 22, 24 and 26 are connected to a public switch 10. The public switch 10 is connected to the company's private branch exchange (PBX) 50. Figure 1 also illustrates a call center 150 comprising a number of agent telephones 152, 154, 156 and 158. The call center 150 is serviced by an automatic call distribution (ACD) system 100. The ACD system 100 comprises a dynamic skills-based router 110 and a set of ACD switching circuits 120. The ACD system 100 routes calls to the agent telephones 152, 154, 156 and 158 according to the method of the present invention. Figure 1 also illustrates a number of telephones 62, 64, 66 and 68 that are used by employees of the company that do not work in the call center 150.

Figure 2 is a functional block diagram of the ACD system 100 and the call center 150 of Figure 1. Again, the ACD system 100 comprises the dynamic skills-based router 110 and the ACD switching circuits 120, and the call center 150 comprises the agent telephones 152, 154, 156 and 158. As further illustrated in Figure 2, the dynamic skills-based router 110 comprises a transaction requestor 200, a transaction queue 202, a queue controller 204, a skills inventory database 206, a transaction dispatcher 208 and a call control interface 210.

In general terms, a number of telephone agents use the agent telephones 152, 154, 156 and 158 to receive or place telephone calls or transactions between the call center 150 and the customer telephones 20, 22, 24 and 26. For example, a customer may use his home telephone 20 to place a telephone call to the company to order one of the company's products. The customer dials a general sales telephone number for the company. The public switch 10 routes the inbound telephone call to the company's PBX 50 by methods that are well known to a person of skill in the art. The PBX 50 routes all inbound telephone calls that are directed to the general sales telephone number to the ACD system 100 for routing to one of the agent telephones 152, 154, 156 or 158. The dynamic skills-based router 110 performs the method of the present invention to determine to which agent telephone 152, 154, 156 or 158 the inbound telephone call or transaction should be routed. The dynamic skills-based router 110 then controls the ACD switching circuits 120 to route the transaction to the selected agent telephone 152, 154, 156 or 158. The call center agent stationed at the selected agent telephone 152, 154, 156 or 158 receives the inbound telephone call or transaction and processes the request. Thus, in the present example of a customer calling to order one of the company's products, the call center agent that receives the inbound call determines which product the customer wants to order, places the order, and arranges for payment by the customer.

The dynamic skills-based router 110 comprises a computer program. In Figure 1, the dynamic skills-based router 110 is illustrated as being a separate functional unit from the PBX 50, and, in the first actual embodiment of the present invention, the router 110 is physically implemented in a stand-alone personal computer that interfaces with the PBX 50. In this embodiment, the call control interface 210 implements a standard CSTA interface. However, in the preferred embodiment of the present invention, the router 110 is physically implemented within the PBX 50. The transaction requestor 200, the queue controller 204, the transaction dispatcher 208 and the call control interface 210 comprise separate routines of the computer program that implements the router 110. The skills inventory database 206 is a database that may be stored on a hard disk and read into random access memory (RAM) upon execution of the program. The transaction queue 202 is preferably generated in RAM during the execution of the program.

Figure 3 is a flow chart illustrating a method performed by the transaction dispatcher 208. The method begins at an initial block 300. At a decision block 302, the transaction dispatcher 208 receives a message (MSG) from any of three sources, an internal timer, the PBX 50 or from an operator at the stand-alone personal computer. At the decision block 302, the transaction dispatcher 208 determines whether the message is a timer message from the internal timer, a direct enqueue message from the operator or a call control message from the PBX switch 50. The call control interface 210 provides an interface between the router 110 and the PBX 50. Thus, for example, the call control interface 210

receives messages from the PBX 50 and forwards the messages other units within the router 110, such as the transaction dispatcher 208.

If the message is a timer message, the method proceeds to a process block 304. The transaction dispatcher 208 controls one or more internal timers within the stand-alone computer, and the internal timers indicate that set times have elapsed by generating messages to the transaction dispatcher 208. If the message received by the transaction dispatcher 208 is a direct enqueue message, the method proceeds to a process block 320. The operator at the stand-alone computer can initiate transactions by the creation of direct enqueue messages. For example, the operator can create a transaction that requires that a call center agent place a telephone call to a specific customer (ie. an outbound transaction). If the message received by the transaction dispatcher 208 is a call control message, the method proceeds to a decision block 322. The PBX 50 generates call control messages to the transaction dispatcher 208, for example, when the PBX 50 receives an incoming telephone call from a customer.

At the process block 304, the transaction dispatcher 208 requests the queue controller 204 for the first transaction from the transaction queue 202.

At a decision block 306, the transaction dispatcher 208 determines whether the transaction received from the queue controller 204 is a null transaction. The transaction will be a null transaction when the transaction queue 202 is empty. If the transaction is a null transaction, the method proceeds to a terminal block 318. Otherwise, the method proceeds to a decision block 308.

At the decision block 308, the transaction dispatcher 208 determines whether the transaction has been in the queue 202 longer than a first predetermined maximum period of time during which transactions are allowed to remain on the queue 202. If the first predetermined period of time has expired, the method proceeds to a process block 310. Otherwise, the method proceeds to a decision block 312.

At the process block 310, the transaction dispatcher 208 returns the transaction to the process that requested the transaction with a message that the transaction timed out. After the process block 310, the method proceeds to a process block 316.

At the decision block 312, the transaction dispatcher 208 determines whether the transaction has been in the queue 202 longer than a second predetermined period of time after which transactions are considered starved. If the second predetermined period of time has expired, the method proceeds to a process block 314. Otherwise, the method proceeds to the process block 316.

At the process block 314, the transaction dispatcher 208 marks the transaction as a starved transaction.

At the process block 316, the transaction dispatcher 208 requests the next transaction in the queue 202 from the queue controller 204. After the process block 316, the method returns to the decision block 306.

At the process block 320, the queue controller 204 receives the message and generates a new transaction. However, there is no incoming telephone call attached to the new transaction, as the transaction was generated from a direct enqueue message, instead of a call control message. After the process block 320, the method proceeds to a decision block 326.

At the decision block 322, the transaction dispatcher 208 determines whether the PBX 50 is reporting that a new incoming telephone call has arrived. If so, the method proceeds to a process block 324. Otherwise, the method proceeds to a decision block 338.

At the process block 324, the transaction requestor 200 creates a new transaction queue entry and attaches the incoming call to the transaction entry. The transaction requestor 200 utilizes the queue controller 204 to place the transaction entry in the queue 202. As described in greater detail below, the transaction entry contains information about the skill requirements of the incoming telephone call. The transaction requestor 200 can obtain the skill requirements of the incoming telephone call in various ways. For example, the transaction requestor 200 can determine the telephone number that was dialed by the customer and the telephone number of the customer. The telephone number that was dialed provides some skill requirement information if different telephone numbers are specified for calls having different requirements. For example, a first number can be specified for English transactions and another number can be specified for Spanish transactions. The transaction requestor 200 can determine whether the transaction requires English skills or Spanish skills from the telephone number that was dialed. The telephone number of the customer may be relevant, for example, if company internal customers have special skill requirements. In addition, the transaction requestor 200 can obtain skill requirement information by prompting the caller to press specific keys on the telephone to indicate the particular skill requirements of the call. The queue controller 204 also forwards the transaction entry directly to the transaction dispatcher 208 for immediate processing and possible immediate dispatching.

At the decision block 326, the transaction dispatcher 208 refers to a list of available agents to determine whether there are any call center agents available to receive the incoming telephone call. The list of available agents is maintained by the transaction dispatcher 208. If there are no agents available, the method proceeds to a process block 332. Otherwise, the method proceeds to a process block 328.

At the process block 328, the transaction dispatcher 208 attempts to match an available agent to the incoming telephone call. The method utilized by the transaction dispatcher 208 during the process block 328 is described in

greater detail below with reference to Figures 4A, 4B and 5.

At a decision block 330, the transaction dispatcher 208 determines whether the method of the process block 328 has resulted in a suitable match between the available agents and the incoming telephone call. If a suitable agent has been found for the incoming call, the method proceeds to a process block 334. Otherwise, the method proceeds to the process block 332.

At the process block 332, the transaction dispatcher 208 leaves the transaction entry in the transaction queue 202 and advances to a terminal block 336.

At the process block 334, the transaction dispatcher 208 assigns the transaction, with any associated telephone call, to the agent selected during the process block 328. More specifically, the transaction dispatcher 208 causes the call control interface 210 to issue a command to the PBX 50, instructing the ACD switching circuits 120 to rout the telephone call, if any, to the agent telephone 152, 154, 156 or 158 corresponding to the selected agent. The transaction dispatcher 208 also controls the queue controller 204 to delete the corresponding entry in the transaction queue 202. After the process block 334, the method proceeds to the terminal block 336.

At the decision block 338, the transaction dispatcher 208 refers to the list of available agents to determine whether any call center agents have become available to process a transaction. If an agent has not just become available, the method proceeds to a decision block 340. Otherwise, the method proceeds to a decision block 336.

At the decision block 340, the transaction dispatcher 208 determines whether a caller has hung up the telephone before speaking with a call center agent. More specifically, the transaction dispatcher 208 determines from the call control message whether a telephone call has been terminated, and, if so, whether the telephone call was associated with an entry in the transaction queue 202. If a caller has hung up the telephone before speaking with an agent, the method proceeds to a process block 342. Otherwise, the method proceeds to a decision block 344.

At the process block 342, the transaction dispatcher 208 removes the entry in the queue 202 that was associated with the terminated telephone call. After the process block 342, the method proceeds to a terminal block 354.

At the decision block 344, the transaction dispatcher 208 refers to the call control message to determine whether an agent has logged onto the ACD system 100. An agent must log on to the ACD system 100 when beginning a call center shift in order to begin receiving transactions. If an agent has logged on, the method proceeds to a process block 346. Otherwise, the method proceeds to a decision block 348.

At the process block 346, the transaction dispatcher 208 adds the agent that just logged on to the list of available agents. After the process block 346, the method proceeds to the terminal block 354.

At the decision block 348, the transaction dispatcher 208 determines whether an agent has logged off the ACD system 100. If an agent has logged off, the method proceeds to a process block 350. Otherwise, the method proceeds to a process block 352.

At the process block 350, the transaction dispatcher 208 removes the agent that just logged off from the list of available agents. After the process block 350, the method proceeds to the terminal block 354.

At the process block 352, the transaction dispatcher 208 reports that it could not recognize the message. After the process block 352, the method proceeds to the terminal block 354.

At the decision block 356, the transaction dispatcher 208 determines from the queue controller 204 whether there are any transactions on the queue 202. If there is at least one transaction on the queue 202, the method proceeds to a process block 358. Otherwise, the method proceeds to a process block 364.

At the process block 358, the transaction dispatcher 208 attempts to match the available agent to one of the transactions on the queue 202. The method utilized by the transaction dispatcher 208 during the process block 358 is described in greater detail below with reference to Figures 4A, 4B and 5.

At a decision block 360, the transaction dispatcher 208 determines whether a transaction has been found on the queue 202 that is suitable for the available agent. If a suitable transaction has been found, the method proceeds to a process block 362. Otherwise, the method proceeds to the process block 364.

At the process block 362, the transaction dispatcher 208 assigns the selected transaction from the queue 202 to the available agent. Again, more specifically, the transaction dispatcher 208 causes the call control interface 210 to issue a command to the PBX 50, instructing the ACD switching circuits 120 to rout the telephone call from the selected transaction, if any, to the agent telephone 152, 154, 156 or 158 corresponding to the available agent. The transaction dispatcher 208 also controls the queue controller 204 to delete the corresponding entry in the transaction queue 202. After the process block 362, the method proceeds to the terminal block 366.

At the process block 364, the transaction dispatcher 208 does not assign a transaction to the available agent and adds the agent to the list of available agents. After the process block 364, the method advances to the terminal block 366.

The method of Figure 3 ends at one of the terminal blocks 318, 336, 354 or 366. When the method ends, the transaction dispatcher 208 waits to receive a subsequent message. When a subsequent message is received, the method of Figure 3 begins again at the initial block 300.

The present invention attempts to find the best agent for each transaction, based on criteria that are considered

to be most important for specific applications. The preferred embodiment assumes that certain criteria are most important, but other criteria may be considered more important in other specific applications. The criteria that are considered most important in the application of the preferred embodiment are described below. In the development of the present invention, the inventors made the following observations:

1) Each agent possesses a set of skills (agent_skills) for handling transactions, and each to some level of expertise (e.g. agent A may be very skilled at speaking Spanish, and moderately skilled at answering technical questions about product X).

$$\text{Agent_skills} = \{ \text{SA1}, \text{SA2}, \text{SA3}, \dots, \text{SA}_i \}$$

where i is the total number of skills in the set.

In the preferred embodiment of the present invention, the skills inventory database 206 contains a data structure corresponding to agent_skills for each agent in the call center 150. Each data structure indicates the skills and the level of expertise in each skill for the corresponding agent.

2) Each transaction of work requires a particular set of skills (trans_skills) (i.e. it requires that the agent performing that transaction possess the skills in trans_skills to a sufficient level of expertise) (for example, if the transaction is to answer a technical question about product X to a Spanish speaking customer, then expertise in both product X and Spanish will be required). This set of skills can be expressed as:

$$\text{Trans_skills} = \{ \text{ST1}, \text{ST2}, \text{ST3}, \dots, \text{ST}_j \}$$

where j is the total number of skills in the set.

Each entry in the transaction queue 202 contains a data structure corresponding to trans_skills that contains the set of skills and the level of expertise for each skill that is required by the corresponding transaction.

3) At any given time T_1 , there is a set or pool of available agents working in the call center, which we will refer to as agents_pool. However, the agents_pool may not contain any members, if there are no available agents at time T_1 . This set does not include agents who are unable to perform a transaction at time T_1 , regardless of the reason for this inability. Agents_pool is the list of available agents referred to above in connection with the decision block 326.

4) At that same time, T_1 , there is also a set or pool of outstanding, unassigned transactions trans_pool. This set does not include any transactions which are already being processed by agents.

In the preferred embodiment of the present invention, the best agent is determined based on the following assumptions:

1) At any time T_1 , if there exists at least one agent in the pool agents_pool with the necessary skills to work on a given transaction trans_x, in the trans_pool, then there must exist at least one best agent (agent_best), who is not under-skilled to work on the transaction, and who is minimally over-skilled to work on the transaction.

2) Agent_best can be calculated by comparing the trans_skills associated with trans_x, with each of the agent_skills possessed by each of the agents in the agents_pool at time T_1 .

Under these assumptions, the agent_best must not be under-skilled to perform the transaction trans_x. Thus, the preferred embodiment of the present invention preferably avoids the under-skilled agent problems described above. If at time T_1 there are no agents in agents_pool which possess the minimum necessary skills to satisfy the requirements of trans_x, then, by our definition of the best agent, there is no best agent for trans_x at time T_1 (although note that there may be best agents for other transactions in the trans_pool, or that at some other time, T_2 , a best agent may become available in the agents_pool).

Also, the agent_best is preferably the minimally over-skilled of all of the agents in agent_pool. In other words, of all of the agents who are able to work on the transaction, the best agent is the one who has the minimum of extra training and expertise which will be wasted (i.e. that will not be required for that particular transaction). If a particular agent has exactly the level of expertise (i.e. no higher than is required), in exactly the set of skills required to perform trans_x, then the measure of how much extra training and expertise being wasted if that agent handles trans_x, is defined as zero. Thus, the present invention also preferably minimizes the over-skilled agent problem, as described above.

For purposes of the preferred embodiment, the best agent is calculated by comparing the set of skills required for

a transaction, to the set of skills possessed by each available agent. This calculation is described in greater detail immediately below.

For each skill in the set agent_skills for a given agent, we will include a property skill_level, which is a value from 1 to m, which expresses an estimate made by call center management of that agent's expertise in that skill (1 meaning that the agent has the minimal expertise acceptable to call center management in that skill, and m meaning the agent is as skilled as any agent can possibly be). So for example, we will write SA1(5) to denote that the agent has skill_level=5 for the skill SA1.

The set of skills possessed by each agent can then be expressed as:

$$\text{Agent_skills} = \{\text{SA1}(\text{LVL1}), \text{SA2}(\text{LVL2}), \text{SA3}(\text{LVL3}), \dots, \text{SAi}(\text{LVLi})\}$$

where i is the total number of skills in the set.

Similarly, the set of skills required to perform trans_x can be expressed as:

$$\text{Trans_skills} = \{\text{ST1}(\text{LVL1}), \text{ST2}(\text{LVL2}), \text{ST3}(\text{LVL3}), \dots, \text{STj}(\text{LVLj})\}$$

where j is the total number of skills in the set.

In the preferred embodiment, an absolute_agent_best for trans_x is defined as the set of agents for whom the following conditions are true:

1) The set trans_skills is equal to the set agent_skills (i.e. they have the same number of elements, all of which are identical, although the level of expertise may vary).

2) For each skill in trans_skills, STx, when compared to its equivalent skill in agent_skills, SAx, the difference between the skill_levels is zero (i.e. $\text{STx}(\text{LVLx}) - \text{SAx}(\text{LVLx}) = 0$). This condition can be stated more succinctly as follows:

$$\text{The sum of } \text{STx}(\text{LVLx}) - \text{SAx}(\text{LVLx}) = 0$$

Where x ranges from 1 to j.

The set of absolute_agent_best is therefore the set of agents who's skills and associated expertise are an exact match with those required to perform trans_x. With an exact match, we can be sure that the center as a whole will not experience either the under-skilled agent problem or the over-skilled agent problem. In practice however, few matches are likely to be exact, and so the preferred embodiment of the present invention implements a method for comparing agents to each other in order to determine which of them is best able to handle a given transaction. We will refer to this method as skills scoring, and it will be based on calculating how close each agent is to being the absolute_agent_best for a given transaction; in this sense, an agent's skills score (relative to a transaction) can be thought of as the difference between the agent's skills and corresponding levels, and the absolute_agent_best's skills and corresponding levels. In this manner, the problems of over-skilled agents and under-skilled agents can be minimized as much as possible for any given transaction.

As described above, one of the major limitations of current ACD systems is that they treat all agents in an ACD group as if they were equally skilled to perform all transactions delegated to the group. The preferred embodiment of this invention avoids any such limitations by providing the skills inventory database 206 within the switch 50, which contains skills-related information about each agent. Each time an agent selection decision or a transaction selection decision needs to be made, the transaction dispatcher 208 can access the information in the skills inventory database 206 in order to be able to make delegation decisions which minimize the negative effects of the under skilled agent and over skilled agent.

Information about the skills each agent possesses is stored in the skills inventory database 206 in the agent's resume. A resume is a formal listing of the skills that an agent possesses. It describes both the level of ability achieved in each skill (i.e. expertise), as well as the level of preference that call center management has, to have the agent handle transactions that require that particular skill versus transactions which require other skills.

In the preferred embodiment, the skills inventory database 206 is implemented using a relational database model. It is therefore comprised of several tables of definitions (which are generally disclosed in Appendix A below). The tables are as follows:

- 1) Skills table
- 2) Agents table
- 3) Resume_details table

The skills table contains a list of all of the valid values for a "skill" in the particular call center using the preferred embodiment of the present invention. A skill is a job or area of expertise in which an agent has competence or experience. In addition, for the purpose of the preferred embodiment, a skill may be further defined as being unique to an individual or common to various individuals, thereby creating a team. For example, a skill may be defined as an agent's name, say bob_skill. Queuing transactions for particular individuals could then be achieved by requesting bob_skill when enqueueing the transaction. As only Bob has bob_skill, the call is effectively queued for Bob. Similarly, teams may be created by giving various people a skill such as sales. All members of the company's sales organization could in this way be designated as having the skill sales, thereby creating a team.

Teams (in the sense used in this description) provide the advantages of prior art ACD groups, without the limitations of the prior art systems.

The agents table contains a list of all of the agents in the call center. For each agent it records a number of important pieces of information. In the present discussion, we are primarily concerned with how it is used to access agent resumes. This is accomplished by use of the resumecode field. Specifically, associated with each agent in the agents table, is a resumecode field which is a key into the resume_details table. In this manner, a computer program which knows the identifier of an agent, can query the agents table, to find out the identifying key used to look up that agent's skills in the resume_details table (i.e. the resumecode).

For each skill which an agent possesses, there will be one row in the resume_details table, which details the identifier of the skill (to reference back to the skills table), and the agent's resumecode (to reference back to the agents table). In addition, the resume_details table contains the following information:

1) Skill level: the skill level is the level of knowledge or expertise that a particular agent has achieved in a given skill. Skill levels are rated on a scale of 1 to 9, with 9 being the highest (i.e. greatest level of expertise).

2) Skill preference: a skill preference is a relative weighting of call center management's desire to have a particular agent handle transactions requiring a particular skill. Skill preferences can be used to reserve uniquely qualified agents for handling calls that require their particular expertise. Consider the example of an agent who possesses the following three skills: SkillA, SkillB, and SkillC. If many other agents in the center have been well-trained to handle transactions using SkillA and SkillB, but only this agent has SkillC, call center management might place a higher preference on SkillC for this agent. Thus, only when no transactions are enqueued requesting SkillC, will the agent handle calls requesting SkillA or SkillB. Skill preferences are rated on a scale of 1 to 9 with 9 being the highest (i.e. most preferred by call center management for this agent to handle).

3) Excluded flag: the excluded flag denotes whether or not a particular skill is an excluded skill for a particular agent. Excluded skills are skills in an agent's resume that the agent is not permitted to handle under any circumstances. For example, in a commission-based sales organization, it may be necessary to restrict certain individuals from handling certain types of calls (e.g. they may not have proper professional certification, and so selling certain products might be illegal).

Not only is the skills inventory database 206 used to store information about the skills possessed by agents, but it is also used to store information about the skills required by transactions (these are stored in the skill_expression table). For each uncompleted transaction currently being handled by the switch, a skills requirement will be associated with it in the skills inventory database 206. A skills requirement expresses the complete set of skills required to complete a particular transaction, by providing both mandatory and optional skill expressions.

A skill expression is a simple formalism for stating the skills required for a particular transaction, which obeys the following context-free grammar:

50	Skillexpression	-> skillexprelement skillexpression
	Skillexprelement	(skillexprelement) not skillexprelement skillexprelement and skillexprelement skillexprelement or skillexprelement skillname, skilllevel <null>
55	Skillname	-> <a skillname from skills inventory>
	Skilllevel	-> 1 2 3 4 5 6 7 8 9

An example of a skill expression based on this grammar is:

Spanish, 5 and stocks, 5 and bonds, 2

This skill expression expresses a requirement for an agent with a level of expertise of 5 in Spanish, who also has a level of expertise of 5 in the skill of stocks, and who also has a level of expertise of 2 in the skill of bonds. (an alternative embodiment of this invention may allow the character "&" to be substituted for the "and" operator, the character "!" to be substituted for the "or" operator, and the character "!" to be substituted for the "not" operator).

A skills requirement consists of both a mandatory and an optional skill expression, in order to allow call center management the flexibility of directing that the same type of transaction be routed one way during normal conditions, and another way during heavy loading periods. Consider for example a call center which has as a business objective, the desire to answer every inbound call within sixty seconds. Management of such a center may desire to have the matching methods consider a complex skill expression as long as the call hasn't yet waited in queue more than the targeted sixty seconds. After that time, they may value timeliness over closeness of match, and so wish to have the switch use a much simpler, less demanding skill expression. Let us therefore define mandatory and optional skill expressions as follows:

Mandatory skill expression: the mandatory skill expression is a skill expression that describes the skills that are mandatory in order to handle a particular transaction. Mandatory skills are always required and the transaction can never be handled by an agent who does not possess the mandatory skills (i.e. during the evaluation of a skill or preference score, agents without the mandatory skills will be considered by the scoring methods as being not suitable).

Optional skill expression: the optional skill expression is a skill expression that describes skills which call center management would like the agent to have before they handle a particular transaction, but which are not mandatory in order for them to complete it successfully. Optional skills are considered mandatory for a particular transaction until the transaction becomes starved, at which time the optional skill requirements are relaxed (i.e. optional skills are considered during the evaluation of a skill score or preference score, but if an agent does not satisfy the skill requirement, he/she will not be found to be not suitable).

Each time the ACD system 100 adds a new transaction to its existing workload, the transaction is assigned an initial priority, as an indication of this new transaction's relative importance as compared to the existing workload. The priority mechanism used in this method is similar to that of traditional ACD systems, but is used much differently (e.g. only in the case of a tied skill match will priorities be considered by the matching methods). The use of priority levels in the preferred embodiment of this invention is also intended to ensure that transactions can not wait in the queue 202 indefinitely. To achieve this result, firstly there is a mechanism provided for regularly increasing the priority of all queued transactions as the length of time in the queue 202 increases. In other words, after a period of time T , the priority of a transaction X is increased from some initial value P_1 , by an increment i , so that after the first interval, the priority level is $P=P_1+i$, after the second interval $P=P_1+2i$, and so on (each of the values P_1 , T , and i can be set by call center management to quantities that work well in a particular business context). Secondly, call center management can set a specially defined priority level called the starvation threshold; once a transaction has a priority greater than the starvation threshold, it is considered to be starved for attention. When transactions become starved, optional skills are still considered in trying to locate the best possible match, however, they are no longer mandatory. In this manner, a starved call is much more likely to find a suitable agent, and so be processed sooner.

Call center management can indicate to the transaction dispatcher 208 how rigorous the evaluation of the skill requirement should be once a given call becomes starved. The mandlevel flag indicates whether skill levels are mandatory once a transaction becomes starved or not. If this flag is set, then even when a transaction is starved, the agent must have the requested level of skill in the mandatory skills to be eligible to handle the call. If the flag is not set, then the agent must still have the skill, but not necessarily at the required level - a lower level will be accepted.

Finally, we also note that transactions can also be assigned priority classes. Priority classes allow a company to provide different levels of service to different customers. A transaction in a higher priority class will always be assigned ahead of a transaction in a lower priority class (assuming a sufficiently skilled agent is there to accept the transaction). This means that a highly skilled agent may be under-utilized when handling a transaction in a higher priority class, so this capability should be used sparingly to maximize the benefits of the matching methods.

The preferred embodiment of the present invention is preferably able to make workable, timely, agent selection decisions and transaction selection decisions. As an improvement upon current ACD systems, the preferred embodiment of the present invention includes the following four methods for making such decisions, each of which makes use of the skills inventory database 206 in order to minimize the traditional limitations caused by the under-skilled agent and the over-skilled agent problems described above. These methods are:

- 4) P: preference-only
- 5) S: skill-only

6) P/S: preference over skill

7) S/P: skill over preference

These methods are referred to as matching methods, because they are used to match agents to transactions. They are based on the assumption that each agent in the call center has a resume in the skills inventory database 206, and that each transaction of work in the call center (whether it be as a result of an inbound call which is answered by the center, or an outbound call which has been, or is about to be, dialed by the center), has an associated skills requirement (as defined above).

Each of these methods involves the calculation of skill scores and preference scores, which are defined as follows:

Skill score: is the closeness of the match between the skills and their associated levels, as possessed by an agent, and the skills and their associated levels as required to complete a transaction. In other words, the skill score is a numeric measure of how close the agent's resume matches the skill requirement of the call. The closer the skill score is to zero, the better the match. This is in keeping with the discussion above regarding finding the best agent, which states that the skill score is the difference between a particular agent and the absolute_best_agent (since a difference of zero would indicate that a particular agent is an absolute_best_agent). Positive scores indicate an over-qualified agent while negative skill scores indicate an under-qualified agent.

Preference score: is a measure of the preference that call center management has for having a particular agent handle a particular transaction. The higher the preference score. The greater the desire of call center management to have that agent work on that particular transaction over other types of transactions.

These two factors, skill score and preference score, can be combined in four different ways to produce the four unique matching methods listed above, each of which provides a valuable method for making agent selection decisions and transaction selection decisions. These four methods may produce different matching results under different conditions, and the center can choose to use any one of the four at any time, whichever provides them the best results given their business goals.

Under each of the four methods, there are two scenarios to consider:

1) Matching transactions to agents (i.e. making the agent selection decision):

this scenario occurs when a transaction arrives in the call center and one or more agents is available to handle the transaction (see the process block 328 of Figure 3).

2) Matching agents to transactions (i.e. making the transaction selection decision):

this scenario occurs when one or more transactions are waiting in queue and an agent becomes available to handle a transaction (see the process block 358 of Figure 3).

The present invention comprises a preference-only method. This method considers only agent preferences for transactions requiring specific skills. Skill levels are not considered at all. The preference-only method can be thought of as follows:

If a transaction exists in queue requiring skills for which the agent has a higher preference level, the agent will take that transaction over a transaction waiting in queue for which the agent has a lower preference level, and

If a transaction arrives and multiple agents are available to handle the call, the transaction will be assigned to the agent who has a higher preference level for the skills required by the call.

For the agent selection decision, the following steps are performed:

1) Build a temporary list of all available agents (List A).

This list is ordered by the amount of time an agent has been idle. In the case of a tie, the Agent who has been idle the longest will receive transaction X (i.e. if two or more equally skilled Agents are vying for the same transaction, then default to the longest idle agent strategy to choose a single agent from amongst them).

2) Step through the temporary list of agents (List A):

Analyze each agent and compare the agent against the new transaction's skill requirement to determine a preference score for each agent. Disqualify all agents with preference scores less than the agent(s) with the highest preference score from List A. The remaining agents are those tied with the highest preference score, and they comprise a list A2.

3) Select an Agent:

EP 0 740 450 A2

If there are no entries in the list A2, then transaction X remains in queue until other agents become available (i.e. no agent is selected during this step).

If there is only one agent in list A2, then transaction X is assigned to that agent.

If there is more than one agent in list A2, then transaction X is assigned to the agent in list A2 which has been idle the longest.

For the transaction selection decision, the following steps are performed:

1) Build a temporary list of all transactions waiting in queue (List T).

This list is ordered by the priority of the transaction. In the case of a tie (two calls with equal priority) the transaction that has been waiting in queue the longest will be handled first.

2) Step through the temporary list of transactions (List T):

Analyze each transaction and compare it against the single available agent to determine a preference score for each transaction. Disqualify all transactions with preference scores less than the transaction(s) with the highest preference score from list T. The remaining transactions are those tied with the highest preference score, and they comprise a list T2.

3) Assign the agent to the transaction with the highest preference score.

If there are no entries in the list T2, then agent A is not assigned a transaction, and therefore remains available.

If there is only one transaction in list T2, then that transaction is assigned to Agent X.

If there is more than one transaction in list T2, then the one which has been waiting longest in the queue is assigned to Agent A.

The preference-only method can be advantageously used when the following are true:

1) Skill levels are not important to call center management

2) Call center management has strong preferences about which skills each agent should be using to handle transactions.

3) Incoming transactions have varying skill requirements.

Figures 4A and 4B are flow charts illustrating the skill-only method. This method considers only agents' levels of skill. The method does not consider call center management's preferences regarding which skills an agent should use. The skills-only method can be thought of as follows:

If transactions are waiting in queue, and an Agent becomes available, the Agent will be assigned the transaction with which the Agent has the closest skills match.

and

If a transaction arrives and multiple Agents are available to handle the transaction, it will be assigned to the Agent with which the transaction has the closest skills match.

Note that the closest skills match means that the under-skilled agent and over-skilled agent problems described above are substantially minimized by using attempting to find the best agent, by deriving a skills score for each agent, relative to a particular transaction. For example, if a transaction requires a skill level of 3 in a particular skill, and one agent has a skill level of 3 in that skill, and another agent has a skill level of 4 in the same skill, the transaction will be assigned to the agent with the level 3 skill. This will leave the agent with the level 4 skill available to handle any calls that require a higher skill level.

For the agent selection decision, the following steps are performed:

1) Build a temporary list of all available Agents (List A):

This list is ordered by the amount of time an Agent has been idle. In the case of a tie, the Agent who has been idle the longest will receive transaction X (i.e. if two or more equally skilled Agents are vying for the same transaction, then default to the longest idle agent strategy to choose a single agent from amongst them).

2) Step through the temporary list of Agents (List A):

Analyze each Agent's resume and compare it to the new transaction's skill requirement to determine a skill score for each agent. See the description of the calculation of skill scores below. Disqualify all Agents with skill scores less than the Agent(s) with the highest skill score from list A. The remaining Agents are those tied with the highest skill score, and they comprise a list A2.

3) Select an agent:

EP 0 740 450 A2

If there are no entries in the list A2, then transaction X remains in queue until other agents become available (i.e. no agent is selected during this step).

If there is only one agent in list A2, then transaction X is assigned to that agent.

If there is more than one agent in list A2, then transaction X is assigned to the agent in list A2 which has been idle the longest.

For the transaction selection decision, the following steps are performed:

1) Build a temporary list of all transactions waiting in queue (List T).

This list is ordered by the priority of the transaction. In the case of a tie (two calls with equal priority) the transaction that has been waiting in queue the longest will be handled first.

2) Step through the temporary list of transactions (List T):

Analyze each transaction and compare it against the single available Agent to determine a skill score for each transaction. Disqualify all transactions with skill scores less than the transaction(s) with the highest skill score from list T. The remaining transactions are those tied with the highest skill score, and they comprise a list T2.

3) Assign the agent to the transaction with the highest skill score.

If there are no entries in the list T2, then agent A is not assigned a transaction, and therefore remains available.

If there is only one transaction in list T2, then that transaction is assigned to agent X.

If there is more than one transaction in list T2, then the one which has been waiting longest in the queue is assigned to agent A.

The skill-only method can be advantageously used when the following are true:

1) Call center management believes that preference levels are not relevant to how calls are managed in the center.

2) Agents have differing levels of expertise in various skills.

3) Incoming transactions have varying skill requirements.

The present invention also comprises the preference/skill method. The preference/skill method provides two refinements to the preference-only method, as follows :

1) In the case of two or more agents having equal preference scores, their respective skill scores are used to break the ties.

2) Agents must satisfy the skill scoring requirements to handle the call.

For the agent selection decision, the following steps are performed:

1) Build a temporary list of all available Agents (List A).

This list is ordered by the amount of time an Agent has been idle. In the case of a tie, the Agent who has been idle the longest will receive transaction X (i.e. if two or more equally skilled Agents are vying for the same transaction, then default to the longest idle agent strategy to choose a single Agent from amongst them).

2) Step through the temporary list of Agents (List A):

Analyze each Agent and compare the Agent against the new transaction X to determine a preference score for each Agent. Disqualify all Agents with preference scores less than the Agent(s) with the highest preference score from List A. The remaining Agents are those tied with the highest preference score, and they comprise a List A2.

3) Try to select an Agent:

If there are no entries in the List A2, then transaction X remains in queue until other Agents become available (i.e. no Agent is selected during this step).

If there is only one Agent in List A2, then transaction X is assigned to that Agent, and no further processing is required (i.e. do not proceed to step 4) below).

If there is more than one Agent in List A2, then continue to the next step.

4) Step through the list of Agents tied with the highest preference score (List A2):

Analyze each Agent and compare the Agent against the new transaction X to determine a skill score for each agent. Eliminate any Agents who are not qualified to handle the transaction based on the skill score. The remaining Agents comprise the List A3.

5) Select an Agent:

If there are no entries in the List A3, then none of the Agents with the currently highest value for preference score have the necessary skills to handle transaction X. From the List A, remove all of the Agents in List A3, to create a new version of List A. If the number of Agents in the new List A is zero, then there are currently no available Agents capable of working on transaction X, so it remains in queue until other Agents become available (i.e. no Agent is selected during this step). If however, the number of Agents in the new List A is greater than zero, then repeat this method, starting from step 2) above.

If there is only one Agent in List A3, then transaction X is assigned to that Agent.

If there is more than one Agent in List A3, then transaction X is assigned to the Agent in List A3 who has the highest skill score (for this transaction). If several Agents are tied with the highest skill score, then the one which has been idle the longest is selected.

For the agent selection decision, the following steps are performed:

1) Build a temporary list of all transactions waiting in queue (List T).

This list is ordered by the priority of the transaction. In the case of a tie (two calls with equal priority) the transaction that has been waiting in queue the longest will be handled first.

2) Step through the temporary list of transactions (List T):

Analyze each transaction and compare it against the single available Agent A to determine a preference score for each transaction. Disqualify all transactions with preference scores less than the transaction(s) with the highest preference score from List T. The remaining transactions are those tied with the highest preference score, and they comprise a List T2.

3) Try to select a transaction:

If there are no entries in the List T2, then Agent A remains available, and all transactions remain in queue (i.e. no transactions is selected during this invocation of the method). No further processing is required (i.e. do not proceed to step 4) below).

If there is only one transaction in List T2, then Agent A is assigned that transaction. No further processing is required (i.e. do not proceed to step 4) below).

If there is more than one transaction in List T2, then continue to the next step.

4) Step through the list of transactions tied with the highest preference score (List T2):

Analyze the skill requirements of each transaction and compare it against Agent A's resume to determine a skill score for each transaction relative to this Agent. Eliminate any transactions for which the Agent does not qualify, based on the skill score. The remaining transactions comprise a List T3.

5) Assign the Agent to the transaction with the highest preference score:

If there are no entries in the List T3, then none of the transactions with the currently highest value for preference score has a skill requirement which can be satisfied by Agent A. From the List T, remove all of the transactions in List T3, to create a new version of List T. If the number of transactions in the new List T is zero, then there are currently no transactions for which Agent A is a capable Agent, so Agent A remains available. (i.e. no Agent is selected during this step). If however, the number of transactions in the new List T is greater than zero, then repeat this method, starting from step 2) above.

If there is only one transaction in List T3, then that transaction is assigned to Agent X.

If there is more than one transaction in List T3, then the one which has the highest skill score (relative to Agent A) is selected. If several transactions are tied with the highest preference score, then the one which has been waiting longest in the queue is assigned to Agent A.

The preference/skill method can be advantageously used when the following are true:

1) Agents in the center have different levels of expertise in different skills, and call center management has preferences regarding which skills each agent should use.

2) The preferences that call center management has, regarding which skills an agent generally uses to perform his job, is more important for making routing decisions than the exact level of expertise that agents have for the various skills.

3) It is important that agents must still be qualified to handle a transaction, even if call center management has a preference for certain agents for the required skill.

4) Incoming calls require differing levels of expertise in various skills.

Figure 5 is a flow chart illustrating the skill/preference method. The skill/preference method provides the following refinement to the skill-only method, as follows:

1) In the case of two or more Agents with equal skill scores, preference scores will be used to break the ties. For the agent selection decision, the following steps are performed:

1) Build a temporary list of all available Agents (List A):

This list is ordered by the amount of time an Agent has been idle. In the case of a tie, the Agent who has been idle the longest will receive transaction X (i.e. if two or more equally skilled Agents are vying for the same transaction, then default to the longest idle agent strategy to choose a single Agent from amongst them).

2) Step through the temporary list of Agents (List A):

Analyze each Agent's resume and compare it to the new transaction's skill requirement to determine a skill score for each Agent. Disqualify all Agents with skill scores less than the Agent(s) with the highest skill score from List A. The remaining Agents are those tied with the highest skill score, and they comprise a List A2.

3) Try to select an Agent:

If there are no entries in the List A2, then transaction X remains in queue until other Agents become available (i.e. no Agent is selected during this step).

If there is only one Agent in List A2, then transaction X is assigned to that Agent, and no further processing is required (i.e. do not proceed to step 4) below).

If there is more than one Agent in List A2, then continue to the next step.

4) Step through the list of Agents (A2):

Analyze each Agent and compare the Agent against the new transaction X to determine a preference score for each agent. Eliminate any Agents who are not qualified to handle the transaction based on the preference. The remaining Agents comprise the List A3.

5) Select an Agent:

If there are no entries in the List A3, then transaction X remains in queue until other Agents become available (i.e. no Agent is selected during this step).

If there is only one Agent in List A3, then transaction X is assigned to that Agent.

If there is more than one Agent in List A3, then transaction X is assigned to the Agent in List A3 who has the highest preference score (for this transaction). If several Agents are tied with the highest preference score, then the one which has been idle the longest is selected.

For the agent selection decision, the following steps are performed:

1) Build a temporary list of all transactions waiting in queue (List T).

This list is ordered by the priority of the transaction. In the case of a tie (two calls with equal priority) the transaction that has been waiting in queue the longest will be handled first.

2) Step through the temporary list of transactions (List T): Analyze each transaction and compare it against the single available Agent's resume to determine a skill score for each transaction. Disqualify all transactions with skill scores less than the transaction(s) with the highest skill score from List T. The remaining transactions are those tied with the highest skill score, and they comprise a List T2.

3) Try to select a transaction:

If there are no entries in the List T2, then Agent A remains available, and all transactions remain in queue (i.e. no transactions is selected during this invocation of the method). No further processing is required (i.e. do not proceed to step 4) below).

If there is only one transaction in List T2, then Agent A is assigned that transaction. No further processing is required (i.e. do not proceed to step 4) below).

If there is more than one transaction in List T2, then continue to the next step.

4) Step through the list of transactions tied with the highest preference score (List T2):

Analyze the skill requirements of each transaction and compare it against Agent A's resume to determine a preference score for each transaction relative to this Agent. Eliminate any transactions for which the Agent does not qualify, based on the preference score. The remaining transactions comprise a List T3.

5) Assign the Agent to the transaction with the highest skill score.

EP 0 740 450 A2

If there are no entries in the List T3, then Agent A is not assigned a transaction, and therefore remains available.
 If there is only one transaction in List T3, then that transaction is assigned to Agent X.
 If there is more than one transaction in List T3, then the one which has the highest preference score (relative to Agent A) is selected. If several transactions are tied with the highest preference score, then the one which has been waiting longest in the queue is assigned to Agent A.

The skill/preference method can be advantageously used when the following are true:

- 1) Agents in the center have different levels of expertise in different skills, and call center management has preferences regarding which skills each Agent should use.
- 2) The preferences that call center management has, regarding which skills an agent generally uses to perform their job, is less important for making routing decisions than the exact level of expertise that agents have for the various skills.
- 3) It is important that agents must still be qualified to handle a transaction, even if call center management has a preference for certain agents for the required skill.
- 4) Incoming calls require differing levels of expertise in various skills.

Skill Scores are calculated by taking each skill in a skill expression and comparing it against a particular Agent's resume. Skill expressions are evaluated from left to right unless parentheses are used to alter the order of evaluation.

After each skill expression in a skill requirement (mandatory and optional) has been evaluated individually, the final cumulative score, which we refer to as the skill score, is the sum of the mandatory and optional scores. If a Skill score is negative, it is penalized by multiplying it by a penalty factor (e.g. -1000). This is because it is preferred that an Agent be overqualified rather than underqualified in handling the call. Applying the penalty factor serves to put any underqualified Agents after any qualified agents.

The highest skill score then, is the score that is closest to 0, with scores above the absolute value of the penalty factor indicating underqualified Agents and scores below the absolute value penalty factor indicating qualified (or over-qualified) Agents.

The following table summarizes the rules used for scoring individual skills in a skills expression, in order to derive a skill score. Specifically, it demonstrates how to derive a skill score (SS) for a single skill, for a particular Agent:

Agent's Suitability for Transaction X	If Agent doesn't have skill	If Agent has Skill at at least the required level	If Agent has skill but at a lower level than required level	If Agent has skill listed in Resume as "excluded"
State of Transaction X				
Not Starved, Mandatory	Agent Not Suitable	SS=Agent's level of skill in resume minus Requested level	Agent Not Suitable	Agent Not Suitable
Not Starved, Optional	Agent Not Suitable	SS=Agent's level of skill in resume minus Requested level	Agent Not Suitable	Agent Not Suitable
Starved, Mandatory, MandLevel Set	Agent Not Suitable	SS=Agent's level of skill in resume minus Requested level	Agent Not Suitable	Agent Not Suitable
Starved, Mandatory, MandLevel Clear	Agent Not Suitable	SS=Agent's level of skill in resume minus Requested level	SS=Agent's level of skill in resume minus Requested level (may be negative)	Agent Not Suitable

(continued)

Agent's Suitability for Transaction X	If Agent doesn't have skill	If Agent has Skill at at least the required level	If Agent has skill but at a lower level than required level	If Agent has skill listed in Resume as "excluded"
State of Transaction X				
Starved, Optional	SS=zero minus Requested Level (will be negative)	SS=Agent's level of skill in resume minus Requested level	SS=Agent's level of skill in resume minus Requested level (may be negative)	Agent Not Suitable
Skill modified by "NOT" (!) operator	SS=zero	Agent Not Suitable	Agent Not Suitable	SS=zero

If two operands are separated by the AND operator and either operand is "Not Suitable", the result of the AND is "Not Suitable". If two operands are separated by the AND operator and neither operand is "Not Suitable", the result of the AND operation is the sum of the operands. If a score is negative, it is penalized by multiplying it by a penalty factor (e.g. -1000). This is because it is preferred that an Agent be overqualified rather than underqualified in handling the call. Applying the penalty factor serves to put any underqualified Agents after any qualified agents.

If two operands are separated by the OR operator and both operands are "Not Suitable" the result of the OR is "Not Suitable". If two operands are separated by the OR operator and either operand is "Not Suitable" the result of the "OR" is the other operand. If two operands are separated by the OR operator and neither operand is "Not Suitable" the result of the "OR" is the lowest number that is greater than or equal to zero. If a score is negative, it is penalized by multiplying it by a penalty factor (e.g. -1000). This is because it is preferred that an Agent be overqualified rather than underqualified in handling the call. Applying the penalty factor serves to put any underqualified Agents after any qualified agents.

Figure 6 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a skill score for a skill expression without parentheses. The method calculates the skill score (SS) for a given Agent A and a given skill expression, which does not contain parentheses. The inputs to the method include the following:

SkillExp: The character string representation of a skill expression.

A: A reference to a single Agent.

T: A reference to the transaction for which SkillExp is part of the skill requirement.

The method of Figure 6 has the following intermediate results:

Operand1: A skill score derived from a part of a skill expression.

Operand2: A skill score derived from a part of a skill expression.

InOp: A value indicating which logical operation to perform upon the two operands, Operand1 and Operand2. The only acceptable values are "AND" to indicate the AND operation, or "OR" to indicate the OR operation.

The method of Figure 6 has the following outputs:

If, as a result of evaluating the skill expression, Agent A has not been disqualified from handling transaction T (i.e. is still considered suitable), then SS is a defined integer result, containing the skill score.

If Agent A has been disqualified, then this result is returned implicitly, and SS is undefined.

If there is an error in the input (a result which is returned implicitly), then SS is undefined.

Figure 7 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a skill score for a skill expression containing AND operators and OR operators. The method of Figure 7 calculates a skill score (SS) for a given Agent A, given two operands, Operand 1, and Operand2, which are each skill scores, and either the operator AND or the operator OR. The method of Figure 7 has the following inputs:

A: A reference to a single Agent.

Operand1: A skill score derived from a part of a skill expression.

5 Operand2: A skill score derived from a part of a skill expression.

T: A reference to the transaction from which Operand1 and Operand2 are partially derived results (i.e. they represent a partial derivation of this transaction's skill requirement).

10 InOp: A value indicating which logical operation to perform upon the two operands, Operand1 and Operand2. The only acceptable values are "AND" to indicate the AND operation, or "OR" to indicate the OR operation.

The method of Figure 7 has the following outputs:

15 If the result of the logical operation is that the skill score does still indicate that Agent A is suitable to handle the skills inherent in the evaluation of the partial skill expression "Operand1 InOp Operand2", then SS is a defined integer result, containing the skill score.

20 If Agent A is not suited to these skills, then this result is returned implicitly, and SS is undefined.

Figure 8 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a skill score for a specific agent, Agent A, and a specific skill, Skill X. The method of Figure 8 has the following inputs:

25 A: A reference to a single Agent.

X: A reference to a single Skill

30 T: A reference to the transaction from which Skill X has been selected (i.e. X is part of this transaction's skill requirement).

ReqLevel: The requested level of expertise for Skill X, as found in transaction T's skill requirement.

ActualLevel: The actual level of expertise for Skill X that Agent A possesses, as found in Agent A's resume.

35 MandLevel: A flag indicating whether or not the MandLevel feature is turned on in the switch.

The method of Figure 8 has the following outputs:

40 If Agent A is suited to handle Skill X, then SS is a defined integer result, containing the skill score.

If Agent A is not suited to handle Skill X, then this result is returned implicitly, and SS is undefined.

As an example of the method for skill scoring, assume the following two agents are available :

45

Skill	Agent1 Level	Agent2 Level
dos	3	6
English	9	4
French	4	9
OS/2	6	1
hardware	5	2

50

Assume further that the following transactions arrive in the queue:

55

Mandatory Skills	Optional Skills	Starved	Mand Level	Score Agent 1	Score Agent 2
dos,3	none	No	n/a	0	3

(continued)

Mandatory Skills	Optional Skills	Starved	Mand Level	Score Agent 1	Score Agent 2
dos,6	none	No	n/a	N/S	0
dos,6	none	Yes	Yes	N/S	0
dos,6	none	Yes	No	-3	0
dos,3 & os2,4	none	No	n/a	2	N/S
none	hardware,1	No	n/a	4	1
dos,3 & os2,4	hardware,1	No	n/a	6	N/S
dos,1los2,1	none	No	n/a	3	0
french,4	hardware,3 & os2,3	No	n/a	5	N/S

Note that the lowest score that is greater than or equal to zero is the best.

Preference Scores are calculated by taking each skill in the skill expression and comparing it against the Agent's resume. Skill expressions are evaluated from left to right unless parentheses are used to alter the order of evaluation. The following table summarizes how the preference score P, for an individual skill S is calculated:

Agents Suitability for Transaction X	Management doesn't have a preference for Agent to do skill	Management does have a preference for Agent to do skill	If Agent has skill listed in Resume as "excluded"
State of Transaction Xs			
Not Starved, Mandatory	Agent Not Suitable	P=preference level in Agent's Resume	Agent Not Suitable
Not Starved, Optional	Agent Not Suitable	P=preference level in Agent's Resume	Agent Not Suitable
Starved, Mandatory	Agent Not Suitable	P=preference level in Agent's Resume	Agent not Suitable
Starved, Optional	P=zero	P=preference level in Agent's Resume	Agent Not Suitable
Skill modified by NOT (!) operator	P=zero	Agent Not Suitable	P=zero

If two operands are separated by the AND operator and either operand is "Not Suitable", the result of the AND is "Not Suitable". If two operands are separated by the AND operator and neither operand is "Not Suitable", the result of the AND operation is the highest preference of the two operands.

If two operands are separated by the OR operator and if both operands are "Not Suitable", the result of the OR is "Not Suitable". If two operands are separated by the OR operator and if either operand is "Not Suitable", the result of the OR is the other operand. If two operands are separated by the OR operator and if neither operand is "Not Suitable", the result of the OR is the highest preference of the two operands.

Figure 9 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a preference score for a skill expression without parentheses. The method of Figure 9 calculates the preference score (PS) for a given Agent A, given a skill expression which does not contain parentheses. The method of Figure 9 has the following inputs:

SkillExp: The character string representation of a Skill Expression.

A: A reference to a single Agent.

T: A reference to the transaction for which SkillExp is part of the Skill Requirement.

The method of Figure 9 has the following intermediate results:

Operand1: A Preference Score derived from a part of a Skill Expression.

Operand2: A Preference Score derived from a part of a Skill Expression.

InOp: A value indicating which logical operation to perform upon the two operands, Operand1 and Operand2. The only acceptable values are "AND" to indicate the "AND" operation, or "OR" to indicate the "OR" operation.

The method of Figure 9 has the following outputs:

If as a result of evaluating the Skill Expression, Agent A has not been disqualified from handling transaction T (i.e. is still considered suitable), then PS is a defined integer result, containing the Preference Score.

If Agent A has been disqualified, then this result is returned implicitly, and PS is undefined.

If there is an error in the input (a result which is returned implicitly), then PS is undefined.

Figure 10 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a preference score for a skill expression containing AND operators and OR operators. The method of Figure 10 calculates the preference score (PS) for a given Agent A, given two operands, Operand 1, and Operand2, which are each preference scores, and either the Operator AND or the Operator OR. The method of Figure 10 has the following inputs:

A: A reference to a single Agent.

Operand1: A Preference Score derived from a part of a Skill Expression.

Operand2: A Preference Score derived from a part of a Skill Expression.

T: A reference to the transaction from which Operand 1 and Operand 2 are partially derived results (i.e. they represent a partial derivation of this Transaction's Skill Requirement).

InOp: A value indicating which logical operation to perform upon the two operands, Operand1 and Operand2. The only acceptable values are "AND" to indicate the "AND" operation, or "OR" to indicate the "OR" operation.

The method of Figure 9 has the following outputs:

If the result of the logical operation is that the Preference Score does still indicate that Agent A is suitable to handle the skills inherent in the evaluation of the partial Skill Expression "Operand1 InOp Operand2", then PS is a defined integer result, containing the Preference Score.

If Agent A is not suited to these skills, then this result is returned implicitly, and PS is undefined.

Figure 11 is a flow chart illustrating a method of the preferred embodiment of the present invention for calculating a preference score for a specific agent and a specific skill. The method of Figure 11 calculates the preference score (PS) for a given Agent A, and a single Skill X. The method of Figure 11 has the following inputs:

A: A reference to a single Agent.

X: A reference to a single Skill

T: A reference to the transaction from which Skill X has been selected (i.e. X is part of this Transaction's Skill Requirement).

ActualPreference: The actual level of preference for Skill X that Call Center Management has associated with Agent A, as found in Agent A's resume.

MandLevel: A flag indicating whether or not the MandLevel feature is turned on in the Switch.

The method of Figure 11 has the following outputs:

If Agent A is suited to handle Skill X, then PS is a defined integer result, containing the Preference Score.

EP 0 740 450 A2

If Agent A is not suited to handle Skill X, then this result is returned implicitly, and PS is undefined.

As an example of the method of calculating preference scores, assume the following two Agents are available:

Skill	Agent1 Level	Agent2 Level
dos	3	6
english	9	4
french	4	9
os2	6	1
hardware	5	2

Assume further that the following calls arrive in the queue :

Mandatory Skills	Optional Skills	Starved	Mand Level	Score Agent 1	Score Agent 2
dos,3	none	No	n/a	3	6
dos,6	none	No	n/a	3	6
dos,6	none	Yes	Yes	3	6
dos,6	none	Yes	No	3	6
dos,3 & os2,4	none	No	n/a	6	6
none	hardware,1	No	n/a	5	2
dos,3 & os2,4	hardware,1	No	n/a	6	6
dos,1los2,1	none	No	n/a	6	6
french,4	hardware,3 & os2,3	No	n/a	6	9

In this example, the highest score is best.

The following two examples illustrate how the four different methods of Figures 4-7 can produce different results for the same calls and the same Agents:

Example 1: Agents available, transaction arrives in queue

Available agents:

	Agent1		Agent2		Agent3		Agent4	
Skill	Level	Pref	Level	Pref	Level	Pref	Level	Pref
SPAN	2	3	4	1	5	2	4	2
STOCKS	2	2	3	1	3	3	3	1

Assume that a call arrives requesting : SPAN,4 & STOCKS,3. The following table provides skill scores, preference scores and idle times:

	Skill Score	Preference Score	Idle Time (s)
Agent1	Not Suitable	3	30
Agent2	0	1	25
Agent3	1	3	20
Agent4	0	2	10

The following table provides the result for each of the methods of Figures 4-7:

Method	Result
Preference	Agent1

(continued)

Method	Result
Skill	Agent1 and Agent3 have equal Preference Scores. Agent1 will get the transaction because he has been idle the longest. Agent2
Preference/ Skill	Agent2 and Agent4 have equal skill scores. Agent2 will get the transaction because Agent2 has been idle the longest. Agent3
Skill/ Preference	Agent1 and Agent3 have equal Preference Scores. Agent3 has a better skill match so he will get the call. Agent4
	Agent2 and Agent4 have equal skill scores. Agent4 has a higher preference match so Agent4 will get the call.

Example 2: Calls in Queue, Agent becomes available

Calls in queue:

Call	Requested Skills
1	SPAN,2
2	TENNIS,4
3	SPAN,5
4	SURFING,3

Assume that Agent1 becomes available with the following resume:

Skill	Level	Pref
SPAN	6	3
SURFING	3	2
TENNIS	4	1

The following table provides skill scores, preference scores and times for which the calls have been in the queue(s).

	Skill Score	Preference Score	Time in queue (s)
Call1	4	3	30
Call2	0	1	20
Call3	1	3	15
Call4	0	2	10

The following table provides the result for each of the methods of Figures 4-7:

Method	Result
Preference	Call1 Call1 and Call3 have equal Preference Scores. Agent will take Call1 as it has been in queue the longest.
Skill	Call2 Call2 and Call4 have equal skill scores. Agent will take Call2 as it has been in queue the longest.
Preference/ Skill	Call3 Call1 and Call3 have equal Preference Scores.

(continued)

Method	Result
Skill/	Agent will take Call3 as it has the better skills match.
Preference	Call4 Call2 and Call4 have equal skill scores. Agent will take Call4 as it has a higher Preference Score.

A person of skill in the art will understand that the present invention can be implemented in various ways that are significantly different from the preferred embodiment.

Skills and Preference Match Scores: The Skills Match and Preference Match Scores provide Call Center managers with a quantitative means to address the "quality of service" and "agent skill utilization" problems discussed above. Viewed from the perspective of the quality of service perspective by the call center overall or individual agents, they enable the call center manager to evaluate how closely they met a caller(s) requirements for a particular transaction (s). Viewed from an agent utilization perspective, they enable call center managers to evaluate how effectively they are utilizing their agents skills (eg: are they over or under qualified) and whether their agents are handling the preferred type of transaction(s) or not.

Skills Match Score: As defined in above for Skill Score, Skill Score is an indicator of the "closeness of the match" or a numeric measure of how close an Agent's Resume matches the Skill Requirement of the call. As with the Skill Score calculated for a call, the Skills Match Score is an indicator of the how over or under-qualified an agent is to handle a call.

Skills Match Score: The Skills Match is the average difference between the skill levels requested by callers and the skill levels of the agents assigned to the calls, for calls that were matched using the Skill Only, Skill Preference, or Preference Skill Methods. It consists of several values:

Skills Match Under-Skilled - The average amount agents were under-skilled.

Skills Match Over-Skilled - The average amount agents were over-skilled.

Before a call is starved it only has a Skills Match Over-Skilled value because the call is only assigned to agents that have the required skill level (or higher) for all skills required by the caller. After a call is starved, both values are provided. Each call can produce both over and under skilled values, as a caller usually requires more than one skill. The agent can be over skilled for one skill and under-skilled for another.

Method of Calculating Skills Match Normal Scores: Use the methods defined above on calculating individual skill scores to calculate SS for each skill in the skills expression for each call that was answered before becoming starved.

Calculate a Skills Match Over-Skilled value (SMOS) for each call by summing the individual skills score SS that are greater than zero. Calculate the average SMOS by summing the SMOS for the number of starved calls m and dividing by m.

Method of Calculating Skills Match Starved Scores: Use the methods defined above on calculating individual skill scores to calculate SS for each skill in the skills expression for each call that was answered after becoming starved.

Calculate a Skills Match Under-Skilled value (SMUS) for each call by summing the individual skill scores SS.

Calculate a Skills Match Over-Skilled value (SMOS) for each call by summing the individual skills score SS that are less than zero.

Display the Individual Skills Match Score in the following format: SMUS, SMOS.

Calculate the average SMUS by summing the SMUS for the number of normal calls n and dividing by n.

Calculate the average SMOS by summing the SMOS for the number of normal calls n and dividing by n.

Display the Skills Match Score in the following format: Average SMUS, Average SMOS.

Example of Calculating Skills Match Starved Scores

For example, assuming that the Skill, Preference matching algorithm is utilized and the call is starved, so the agent does not have to have the required skill levels to handle the call.

A call is enqueued with the following skill expression:

SPANISH,3 & PRODUCT,5 & SERVICE,7

The call is routed to an agent with the following resume:

EP 0 740 450 A2

Skill	Level	Preference
SPANISH	2	3
PRODUCT	8	7
PRODUCT B	5	5
SERVICE	8	8

The individual Skills Score (SS) is calculated according to the methods defined above on Calculating Skill Scores. The skill levels required by the caller is subtracted from the skill level provided by the agent for each of the skills in the caller's expression. The results are as follows:

For SPANISH, SS=-1

For PRODUCT, SS=+3

For SERVICE, SS=+1

To calculate the individual Skills Match Over-Skilled, the individual skills scores (SS) that are greater than zero are summed. To calculate the individual Skills Match Under-Skilled, the individual skill scores (SS) that are less than zero are summed. In this example, Skills Match Under-Qualified = -1 and Skills Match Over-Qualified = 4. The Skills Match values for all calls are calculated in the same way and then are averaged together.

The Skills Match values can be used to determine how well your call center is matching agents to calls. The values 0,0 are the best possible values indicating that the caller was handled by Agent_Best as defined above, Theory for Determining the "Best" Agent.

If you have a high negative number, you are not meeting the skill requirements of your callers. This may indicate that you need to train agents to increase their skill levels. If you have a high positive number, your agents are highly overqualified for your callers' requirements, which could mean that most of your skilled agents are on one team or shift.

Preference Match Scores: As discussed in the section on Preference Scores, Preference Score is a measure of the "preference" that a Call Center Manager has for having a particular agent handle a particular transaction. The higher the Preference Score, the greater the desire of a Call Center Manager to have the agent work on that particular transaction over other types of transactions.

Preference Match: Preference Match is the average Preference Score that agents had for calls that were assigned to them, when calls were matched using the Preference Only, Preference Skill, or Skill Preference Matching Methods. As with Preference Score, the higher the Preference Match value the more successful the application was at matching calls to agents based on preferences. The Preference Match Score consists of two values.

Preference Match Normal - The average preference score that agents had for calls that were assigned to them before the calls became starved. Preference Match Starved -The average preference score that agents had for calls that were assigned to them after the calls became starved.

Method of Calculating Preference Match Normal:

1. For all calls that were answered before being starved, calculate PS as defined in section on Methods of calculating Preference Scores.
2. Calculate the average Preference Match Normal value by summing the PS for the number of normal calls n answered and dividing by n.

Method of Calculating Preference Match Starved:

1. For all calls that were answered after being starved, calculate PS as defined in section on Methods of calculating Preference Scores.
2. Calculate the average Preference Match Starved value by summing the PS for the number of starved calls m answered and dividing by m.

APPENDIX A

AGENTHISTORY Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
IntervalStartA	NO	DATETIME	PK
IntervalEndA	NO	DATETIME	PK
AgentID	NO	CHAR 8	PK
RoundedStartA	NO	DATETIME	
RoundedEndA	NO	DATETIME	
Avail_R		LargeInt	
Unavail_R		LargeInt	
Work_R		LargeInt	
Pend_R		LargeInt	
OnCall_R		LargeInt	
OnHold_R		LargeInt	
ACD_R		LargeInt	
Unknown_R		LargeInt	
Other_R		LargeInt	
NonMMCall_R		LargeInt	

AGENTS Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
Userid	NO	CHAR 8	
Password	NO	CHAR 8	
Usertype	NO	SMALLINT	
AutoLogon	NO	SMALLINT	
AutoWork	NO	SMALLINT	
CallPriveleges	NO	SMALLINT	
AgentID	NO	CHAR 8	PK
LastName	NO	CHAR 35	
MidInitial		CHAR 5	
FirstName	NO	CHAR 30	
Availability	NO	SMALLINT	
Extension	NO	CHAR 32	
ResumeID		CHAR 8	FK (Resumes)
ACDGroup	NO	CHAR 32	
ACDGroupNm		CHAR 32	
Department		CHAR 30	
Comment		CHAR 80	
BackupID		CHAR 8	FK (Agents)
BackupStartDate		TIMESTAMP	
BackupEndDate		TIMESTAMP	
Comment		CHAR 80	

CALL_HISTORY Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
Timestamp	NO	DATETIME	PK
TADS Callid count	NO	LargeInt	
TADS Callid time	NO	LargeInt	
TADS Callid server	NO	LargeInt	
Call Type	NO	SmallInt	
Requester ID	NO	CHAR 8	
Requester Extension	NO	CHAR 32	
ANI		CHAR 32	
DNIS		CHAR 32	
Initial Priority	NO	SMALLINT	
Dequeue Priority	NO	SMALLINT	
StarvePriority	NO	SMALLINT	
PriorityClass	NO	SMALLINT	
MaxWait	NO	SMALLINT	
Mand Skills		VARCHAR 1000	
Opt Skills		VARCHAR 1000	
Time Enqueued	NO	DATETIME	
Time Assigned		DATETIME	
Time Dequeued		DATETIME	
Agent ID		CHAR 8	
Agent Extension		CHAR 32	
Call Caption		CHAR 30	
Dequeue Reason	NO	SMALLINT	
Dequeue Comment		CHAR 80	
skillscore		SMALLINT	
prefScore		SMALLINT	

EL_ROUTING Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
SortIndex	NO	CHAR 67	
ANI	NO	CHAR 32	PK
DNIS	NO	CHAR 32	PK
mandSkills		VARCHAR 1000	
optSkills		VARCHAR 1000	
skillexpname		CHAR 30	FK (SkillExps)
priority	NO	SMALLINT	
priorityclass	NO	SMALLINT	
maxwait	NO	SMALLINT	
caption		CHAR (80)	
comment		CHAR (512)	

REQUESTERS Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
RequesterCode	NO	CHAR 8	PK
RequesterName	NO	CHAR 30	
RequesterType	NO	SMALLINT	
StartupExitName		CHAR 12	
CallExitName		CHAR 12	
HBACDNum		CHAR 32	
OrigExt1		CHAR 32	
OrigExt2		CHAR 32	
OrphanACDNum		CHAR 32	
XferCommand		CHAR 10	
Comment		CHAR 80	

EP 0 740 450 A2

REQUESTER_RESOURCES Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
RequesterCode	NO	CHAR 8	PK, FK (Requesters)
Extension	NO	CHAR 32	PK
ACDGroupName		CHAR 32	

RESUMES Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
ResumeCode	NO	CHAR 8	PK
ResumeName	NO	CHAR 30	
Comment		CHAR 80	

RESUME _DETAILS Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
ResumeCode	NO	CHAR 8	PK, FK (Resumes)
SkillCode	NO	CHAR 8	PK, FK (Skills)
SkillLevel	NO	CHAR 1	
Excluded	NO	CHAR 1	
Preference	NO	CHAR 1	
Comment		CHAR 80	

SKILLS_EXPRESSIONS Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
SkillExpressionName	NO	CHAR 30	PK
MandSkillExpression		VARCHAR 1000	
OptSkillExpression		VARCHAR 1000	
Comment		CHAR 80	

SKILLS Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
SkillCode	NO	CHAR 8	PK
SkillName	NO	CHAR 30	
ParentCode		CHAR 8	FK (Skills)
Comment		CHAR 80	

SYSPARMS Table

Field Name	Can Field be NULL?	Datatype	Is Field a Key?
ParmName	NO	CHAR 30	
ParmValue	NO	CHAR 80	
Comment		CHAR 200	

EP 0 740 450 A2

Valid System Parameters

Parameter Name	Range of Valid Values	Description
SwitchType	0,1	9005 or 9006
Passive	0,1	No, Yes
MatchMethod	0-3	ie. BestAgent or Preference
WaitTime	0-3600	When a transaction has waited WaitTime seconds, boost
WaitAmount	0-99	priority by WaitAmount.
ServLevMin	0-99 (future)	When Service Level falls below ServLevMin,
ServLevDif	0-99 (future)	by ServLevDif percentage points, boost
ServLevAmount	0-99 (future)	priority by ServLevAmount.
AbandonMax	0-99 (future)	When Abandon Rate rises above AbandonMax
AbandonDif	0-99 (future)	by AbandonDif percentage points, boost
AbandonAmount	0-99 (future)	priority by AbandonAmount.
MaxAgentPendingTime	0-3600	Max time an Agent can spend in pending state. This parm will depend on the maximum time the VRU will be in an uninterruptable state.
MaxCallPendingTime	0-3600	Max time a transaction can spend in pending state.
MaxCallOrphanTime	0-3600	Max time a transaction can spend in orphan state.
StarvePriority	0-99	Priority at which calls are considered starved.
MandLevel	0,1	Levels on mandatories aren't/are flexible when transaction is starved
TADSServerName		TADS Server Name for MMKR Server
AgentActivityPeriod		
ELRequester	0,1	Y/N
MaxReportDays	0-360	Max number of days of history records to keep.

AgentStatusType		Qviewer or Telephone
MinFreeSpace		Minimum Amount of free space to leave on db drive
QVRefreshAgent		Qviewer Refresh Interval for Agents
QVRefreshSuper		Qviewer Refresh Interval for Supervisors
QVAgentsAgent		Max # Agents Agent can view
QVAgentsSuper		Max # Agents supervisor can view
QVSkillsAgent		Max # skills Agent can view
QVSkillsSuper		Max # Skills supervisor can view
QVCallsAgent		Max # calls Agent can view
QVCallsSuper		Max # calls supervisor can view
QVDSRefresh		Qviewer Data Server refresh rate

Claims

1. An automatic call distribution system for distributing a transaction between a plurality of agents at a call center, said automatic call distribution system comprising:
 - a transaction queue, said transaction queue storing a set of skill requirement data for said transaction;
 - a skills inventory database, said skills inventory database storing a set of skill data for each of said plurality of agents;
 - a transaction dispatcher, said transaction dispatcher comparing said set of skill requirement data for the transaction with said skill data for said agents and selecting one of said agents for processing of said transaction; and
 - a switching circuit, said switching circuit responsive to said transaction dispatcher for switching said transaction to said selected agent.

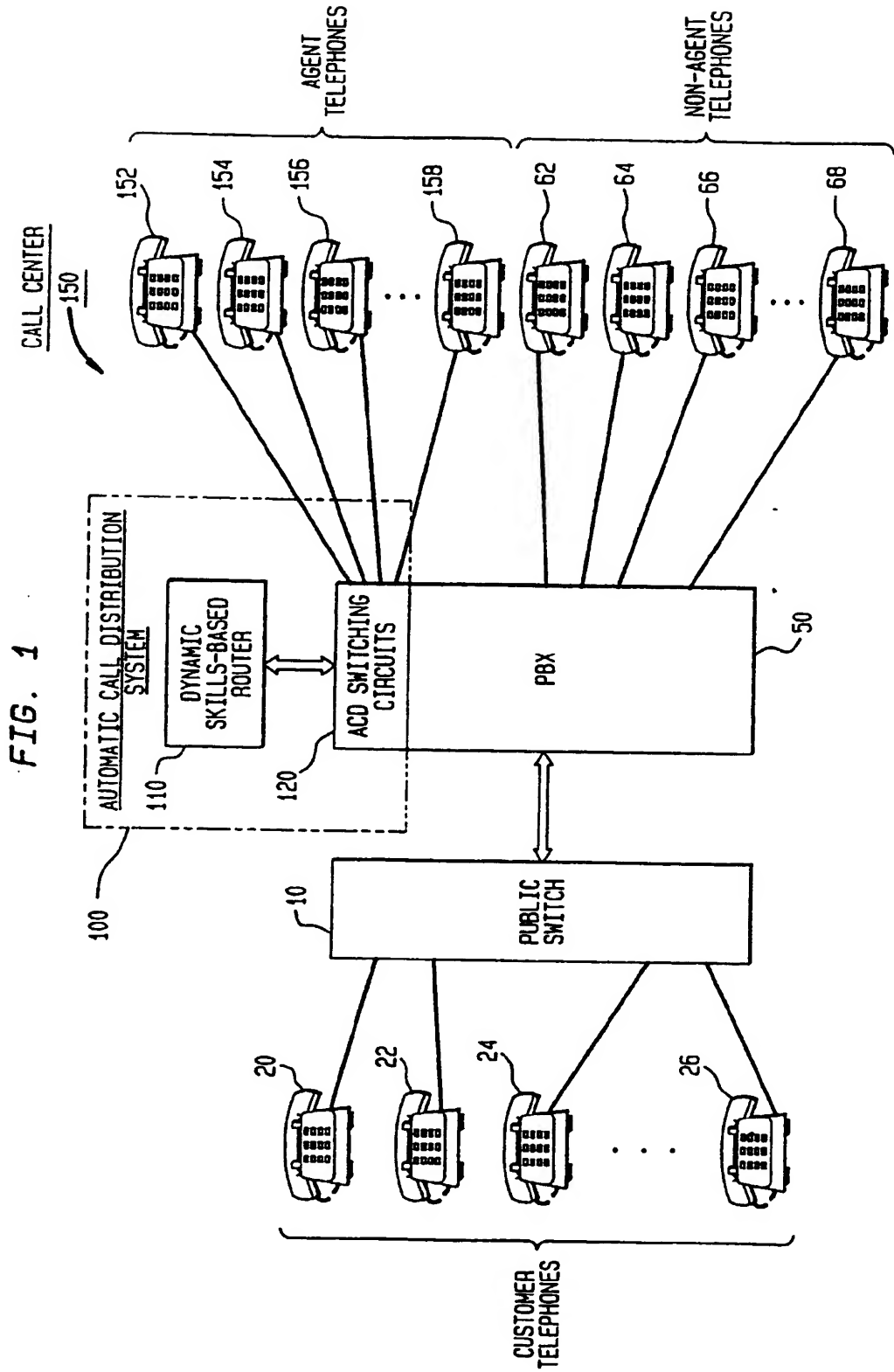


FIG. 2

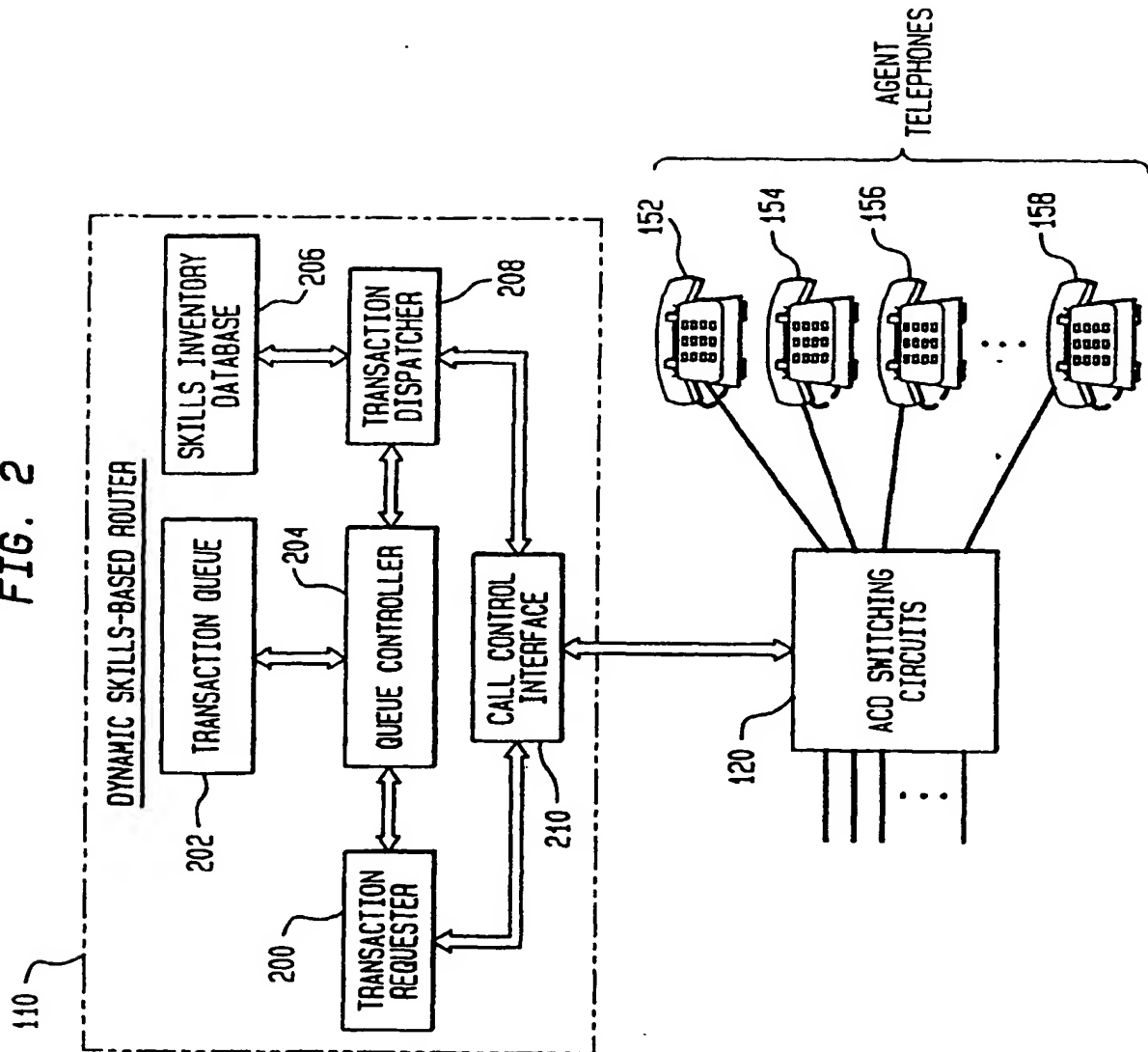


FIG. 3

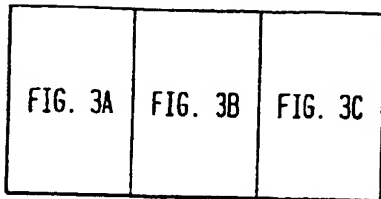


FIG. 3A

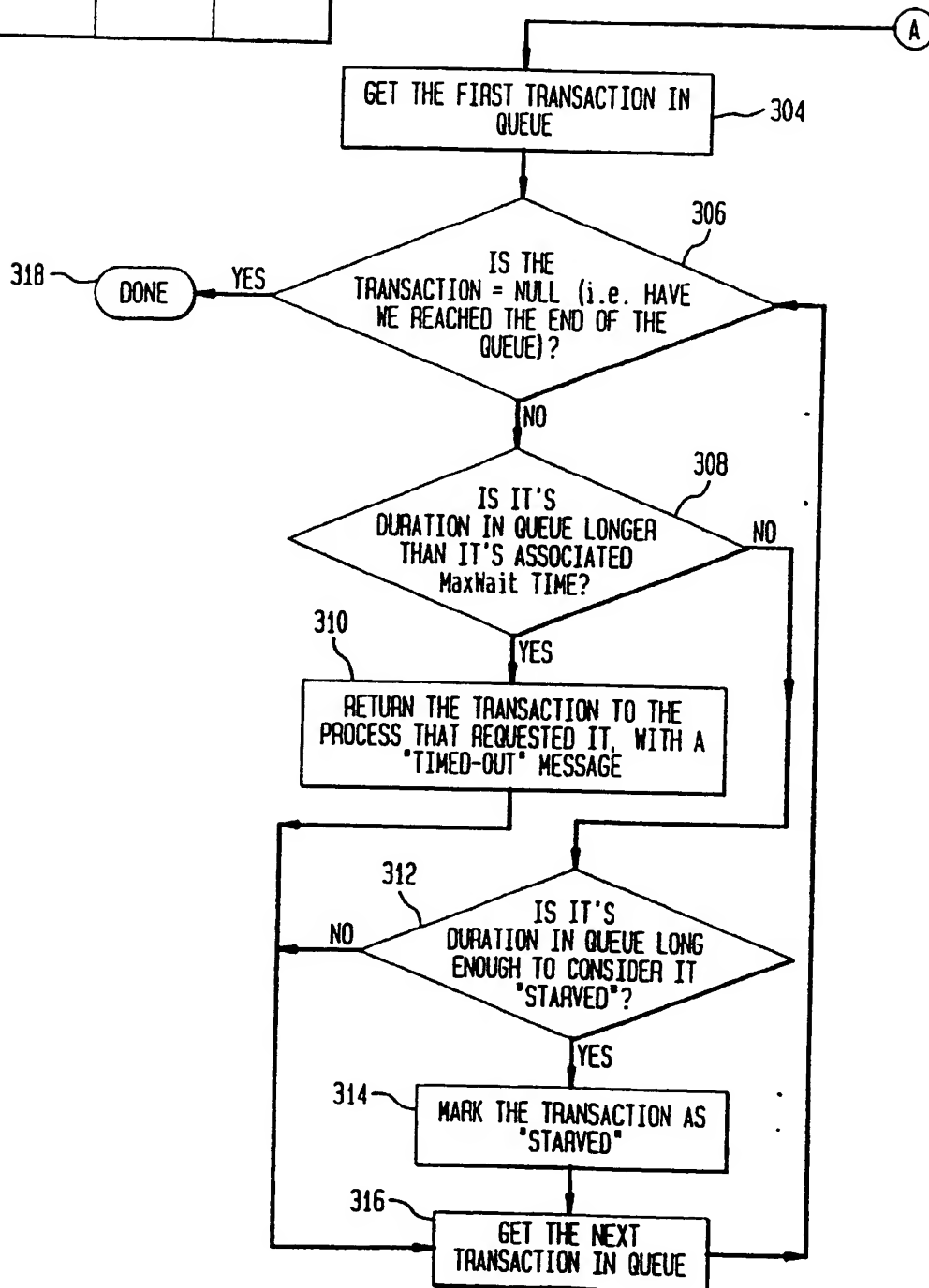


FIG. 3B

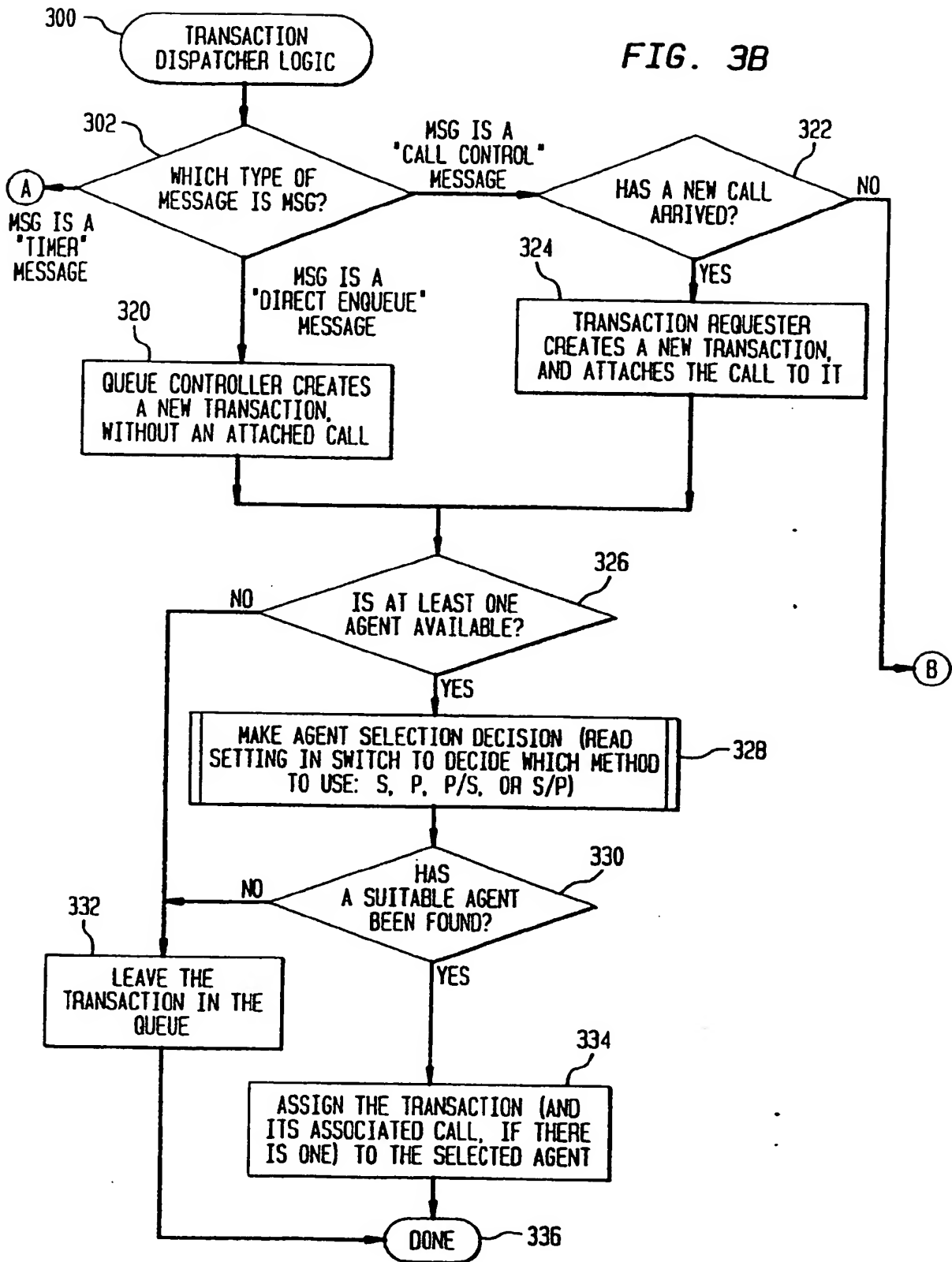
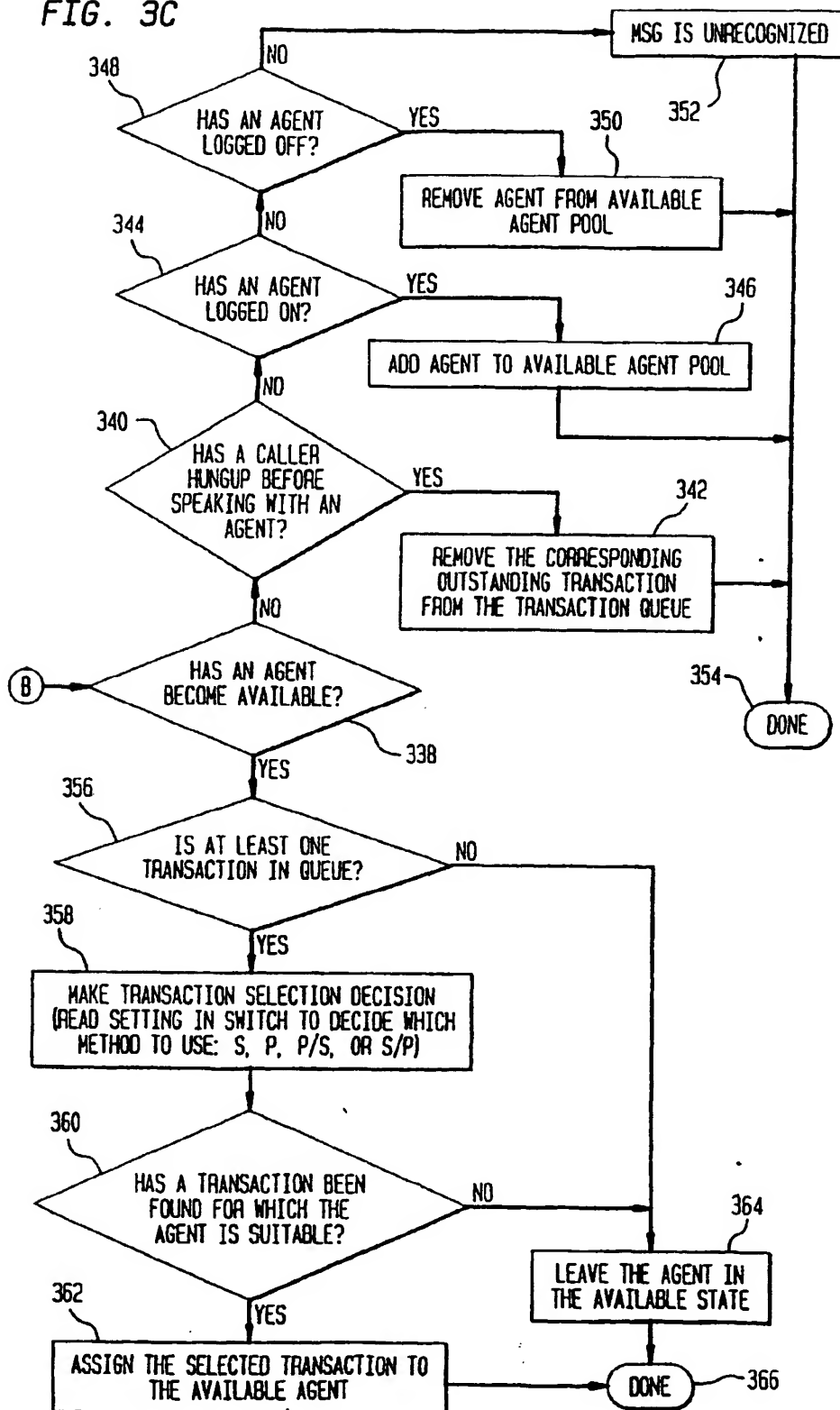
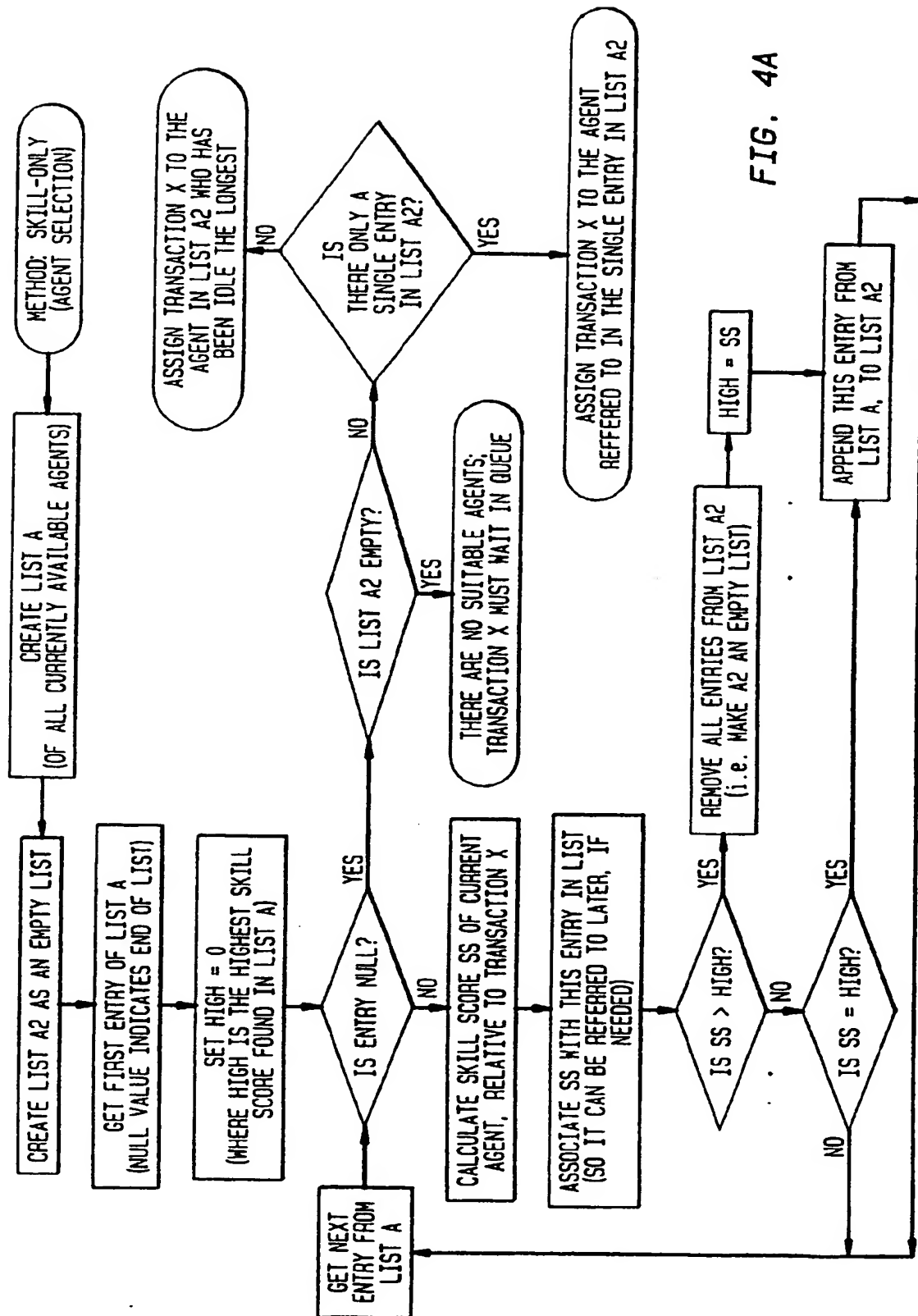


FIG. 3C





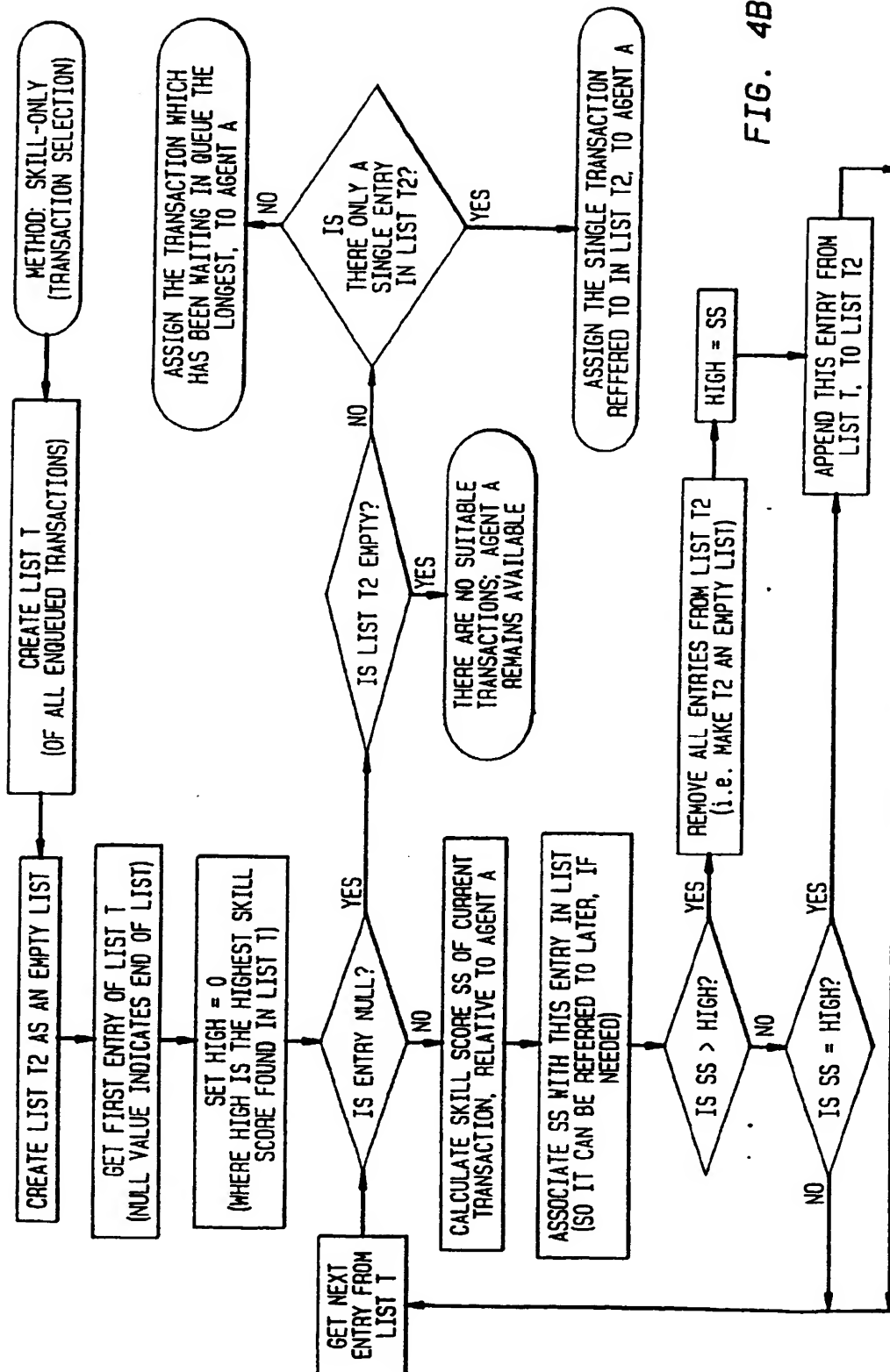


FIG. 5

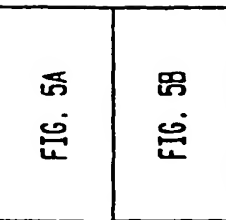
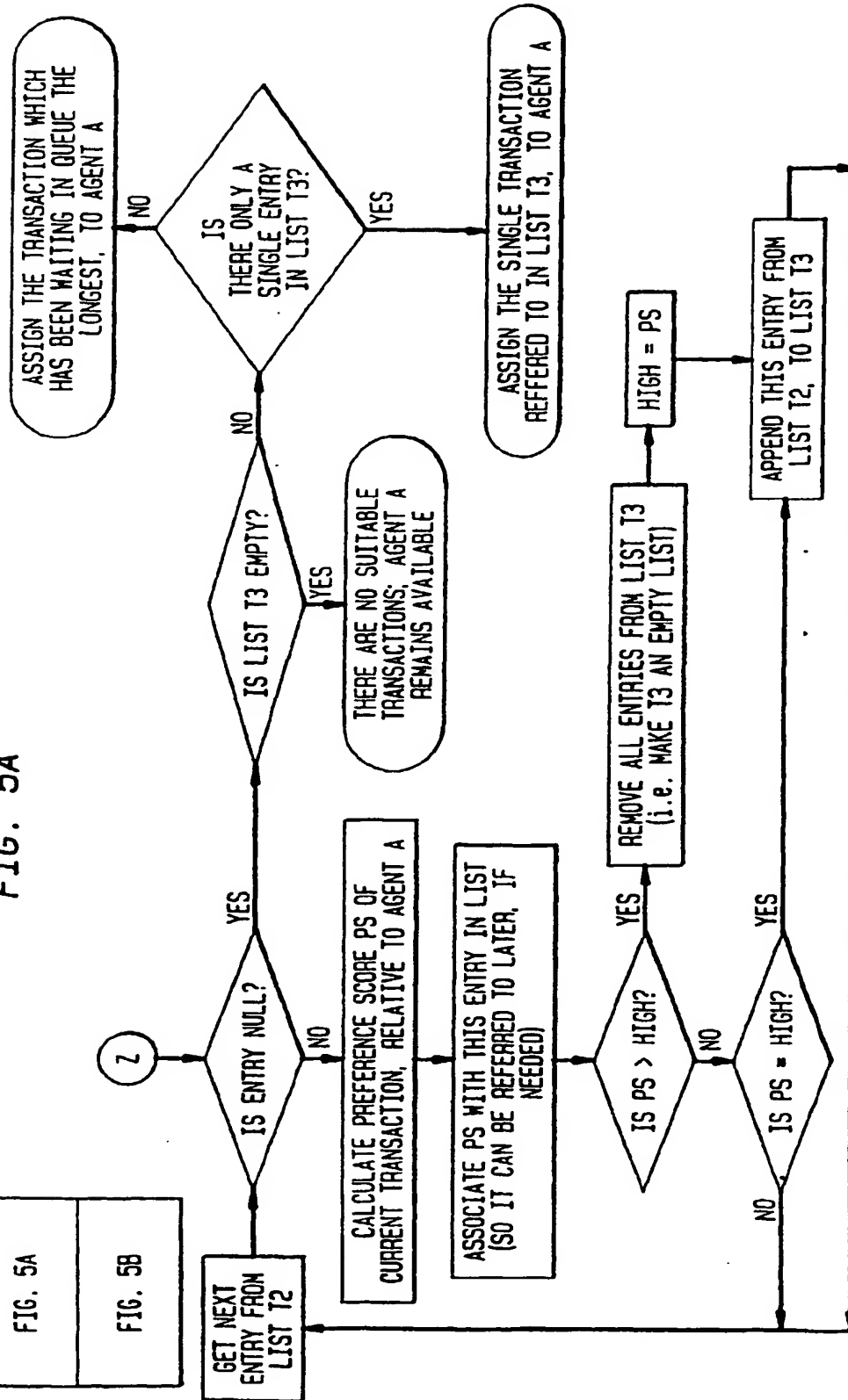


FIG. 5A



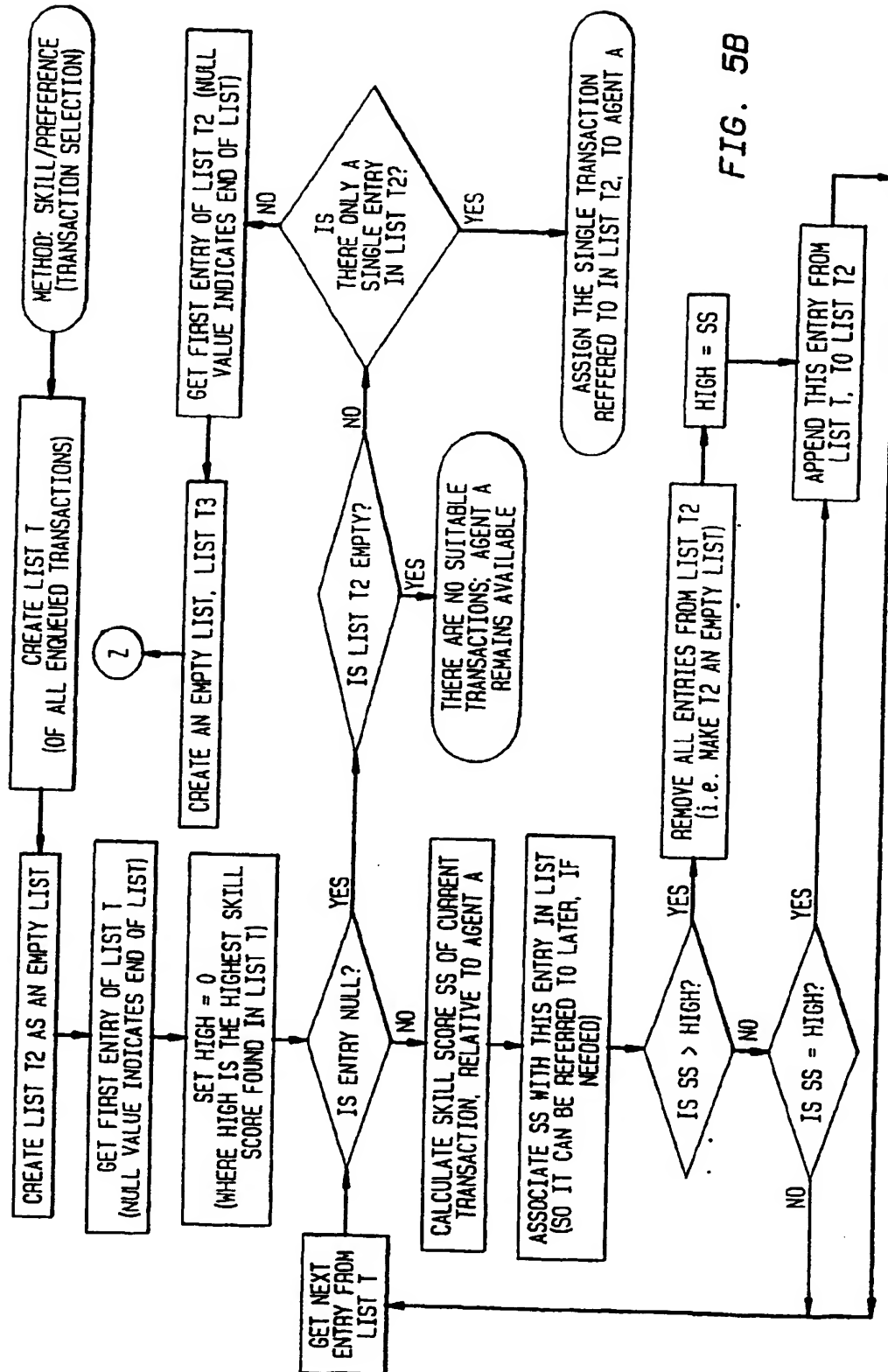


FIG. 5B

FIG. 6A

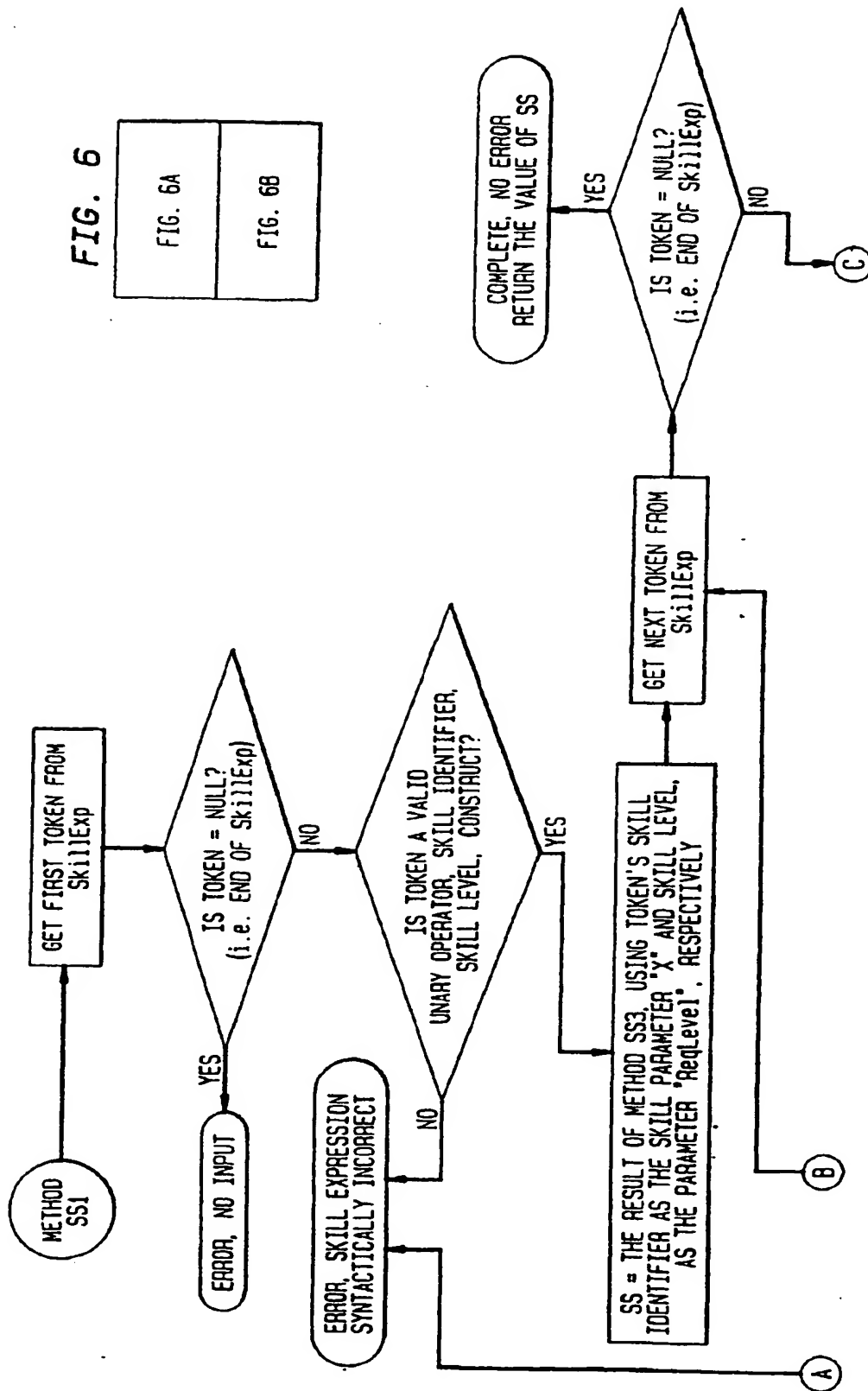
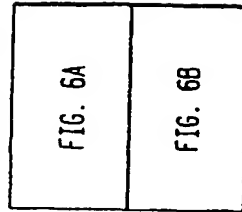


FIG. 6



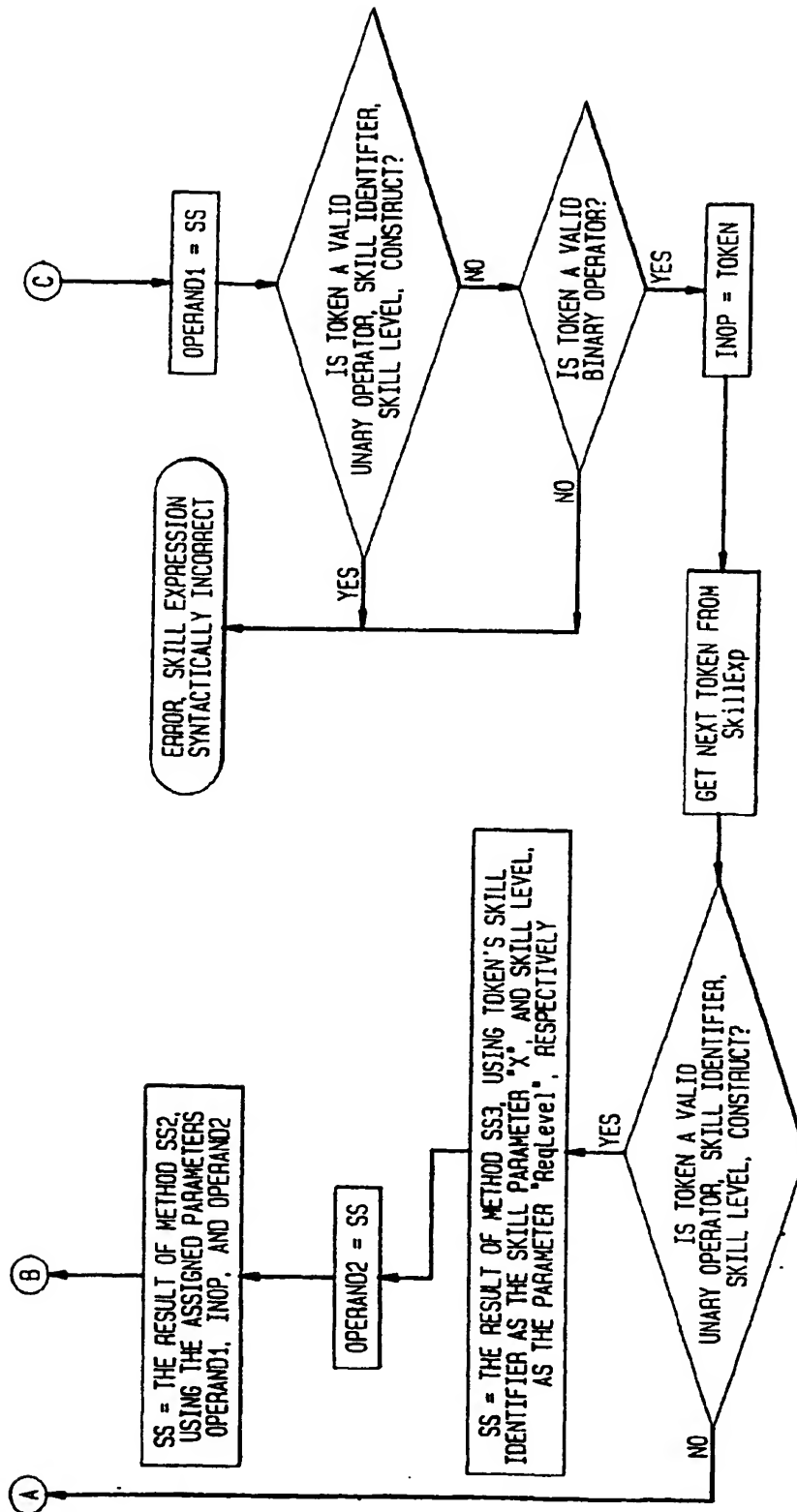


FIG. 6B

FIG. 7

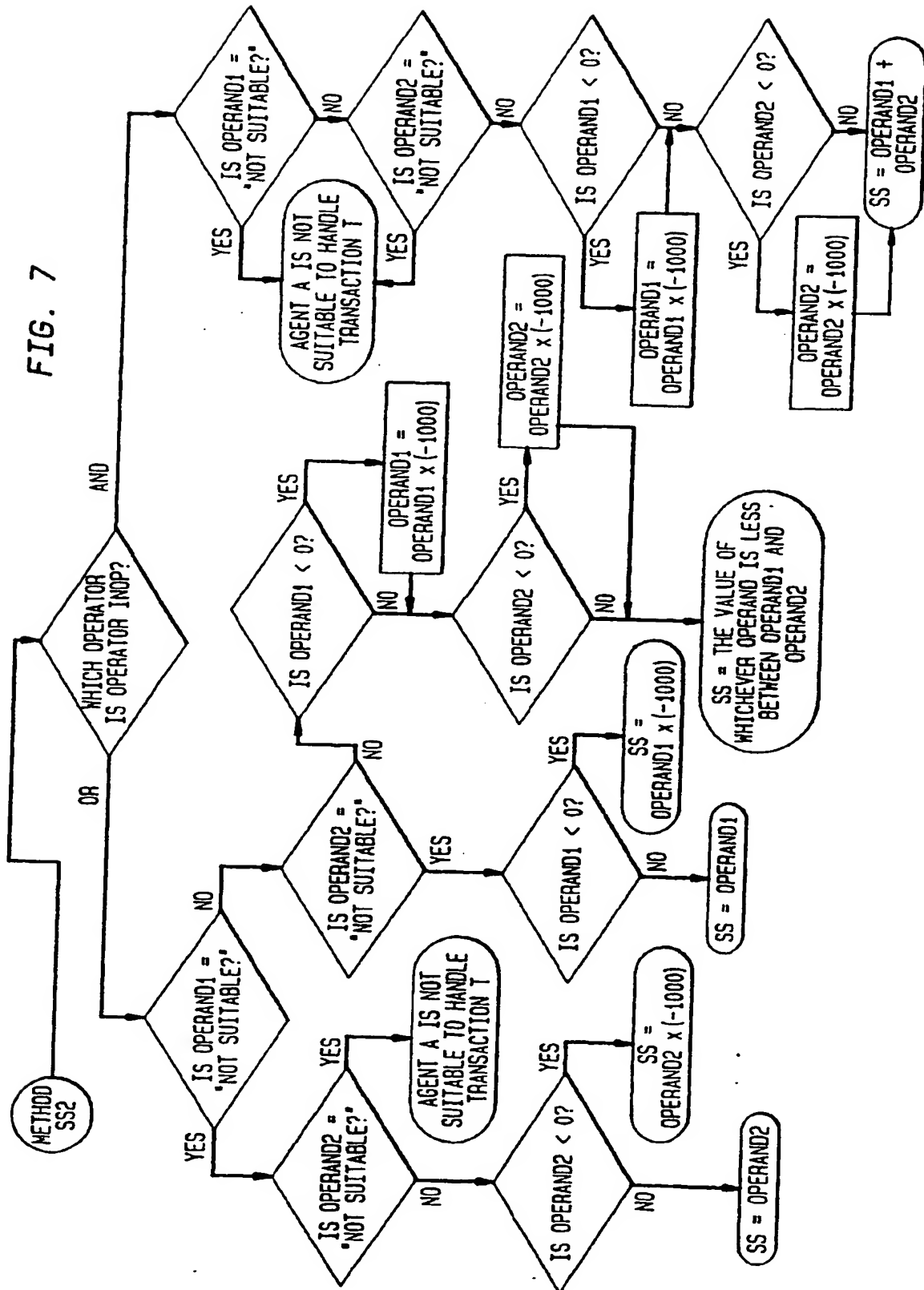


FIG. 8

```
graph TD
    Start([METHOD SS3]) --> D1{IS SKILL X PRECEDED BY "NOT" OPERATOR?}
    D1 -- NO --> D2{IS TRANSACTION T "STARVED"?}
    D1 -- YES --> D3{DOES AGENT POSSESS THIS SKILL?}
    D2 -- YES --> D4{IS SKILL X MANDATORY FOR TRANSACTION T?}
    D2 -- NO --> D5{DOES AGENT A POSSESS SKILL X?}
    D4 -- YES --> D6{IS HandLevel OF SWITCH SET ON?}
    D4 -- NO --> D5
    D6 -- YES --> D5
    D6 -- NO --> D7{DOES AGENT A POSSESS SKILL X?}
    D5 -- YES --> D8{IS THIS SKILL MARKED AS "EXCLUDED"?}
    D5 -- NO --> D9{IS AGENT A'S ActualLevel >= ReqLevel?}
    D8 -- YES --> D9
    D8 -- NO --> D10{IS AGENT A'S ActualLevel >= ReqLevel?}
    D9 -- YES --> E1([AGENT A IS NOT SUITABLE TO HANDLE TRANSACTION T])
    D9 -- NO --> E2([AGENT A IS NOT SUITABLE TO HANDLE TRANSACTION T])
    D10 -- YES --> E3([SS = ActualLevel - ReqLevel])
    D10 -- NO --> E4([IS AGENT A'S ActualLevel >= ReqLevel?])
    E4 -- YES --> E5([SS = ActualLevel - ReqLevel])
    E4 -- NO --> E6([SS = ActualLevel - ReqLevel])
    E6 --> E7([SS = 0 - ReqLevel])
    E7 --> E8([SS = ZERO])
    E8 --> E9([AGENT A IS NOT SUITABLE TO HANDLE TRANSACTION T])
```

The flowchart, labeled FIG. 8, illustrates the logic of Method SS3. It begins with a start node (METHOD SS3) leading to a decision diamond: "IS SKILL X PRECEDED BY 'NOT' OPERATOR?". If the answer is "NO", the flow proceeds to "IS TRANSACTION T 'STARVED'?". If "YES", it goes to "DOES AGENT POSSESS THIS SKILL?". From "IS TRANSACTION T 'STARVED'?", a "YES" leads to "IS SKILL X MANDATORY FOR TRANSACTION T?". If "YES", it checks "IS HandLevel OF SWITCH SET ON?". If "YES", it proceeds to "DOES AGENT A POSSESS SKILL X?". If "NO", it goes directly to "DOES AGENT A POSSESS SKILL X?". From "DOES AGENT A POSSESS SKILL X?", a "YES" leads to "IS THIS SKILL MARKED AS 'EXCLUDED'?". If "YES", it proceeds to "IS AGENT A'S ActualLevel >= ReqLevel?". If "NO", it goes to "IS AGENT A'S ActualLevel >= ReqLevel?". From "IS AGENT A'S ActualLevel >= ReqLevel?", a "YES" leads to "AGENT A IS NOT SUITABLE TO HANDLE TRANSACTION T". A "NO" leads to "IS AGENT A'S ActualLevel >= ReqLevel?". From this diamond, a "YES" leads to "SS = ActualLevel - ReqLevel", and a "NO" leads to "SS = ActualLevel - ReqLevel". From "SS = ActualLevel - ReqLevel", a "YES" leads to "SS = ActualLevel - ReqLevel", and a "NO" leads to "SS = 0 - ReqLevel". From "SS = 0 - ReqLevel", the flow leads to "SS = ZERO", which then leads to "AGENT A IS NOT SUITABLE TO HANDLE TRANSACTION T".

FIG. 9

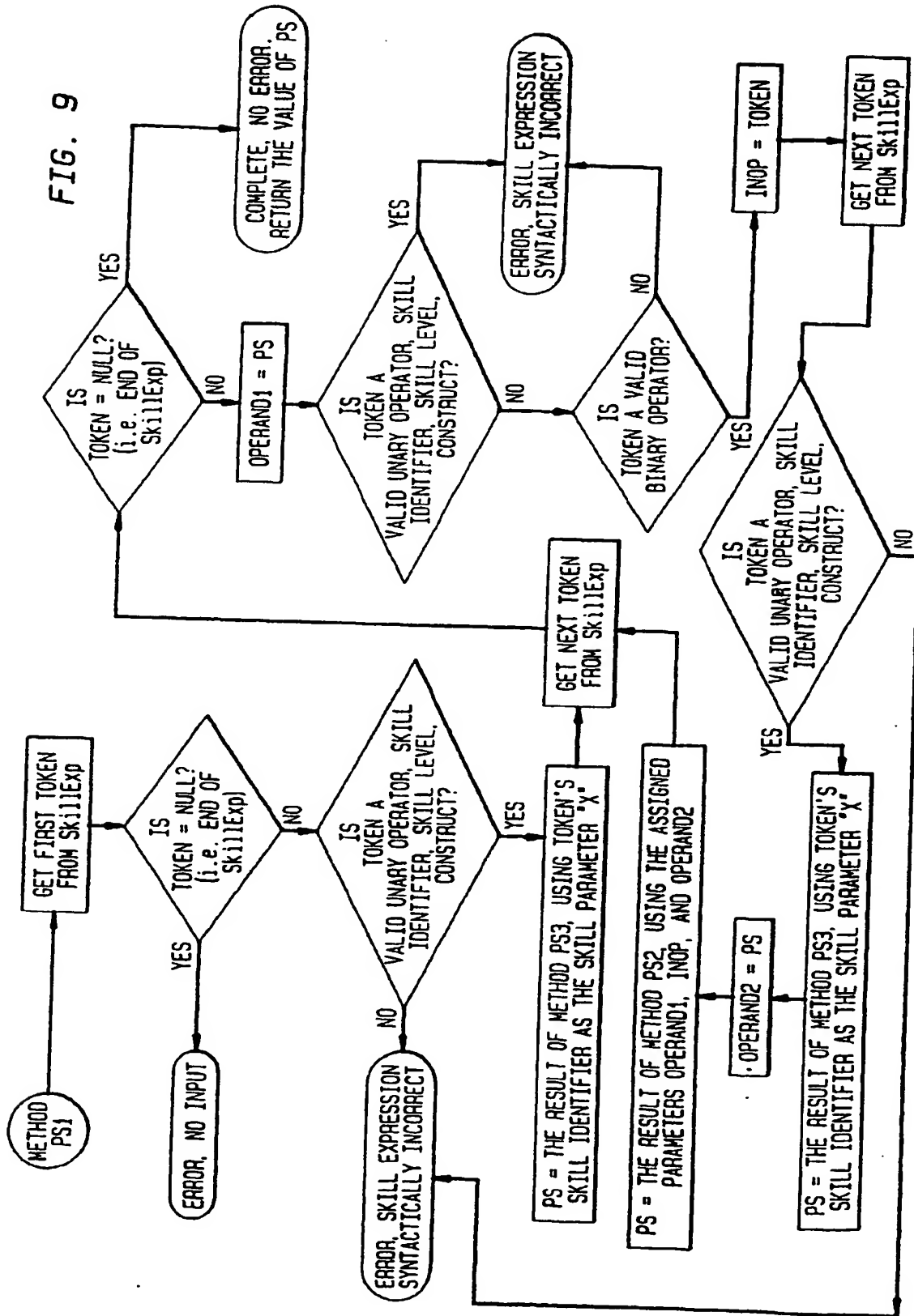


FIG. 10

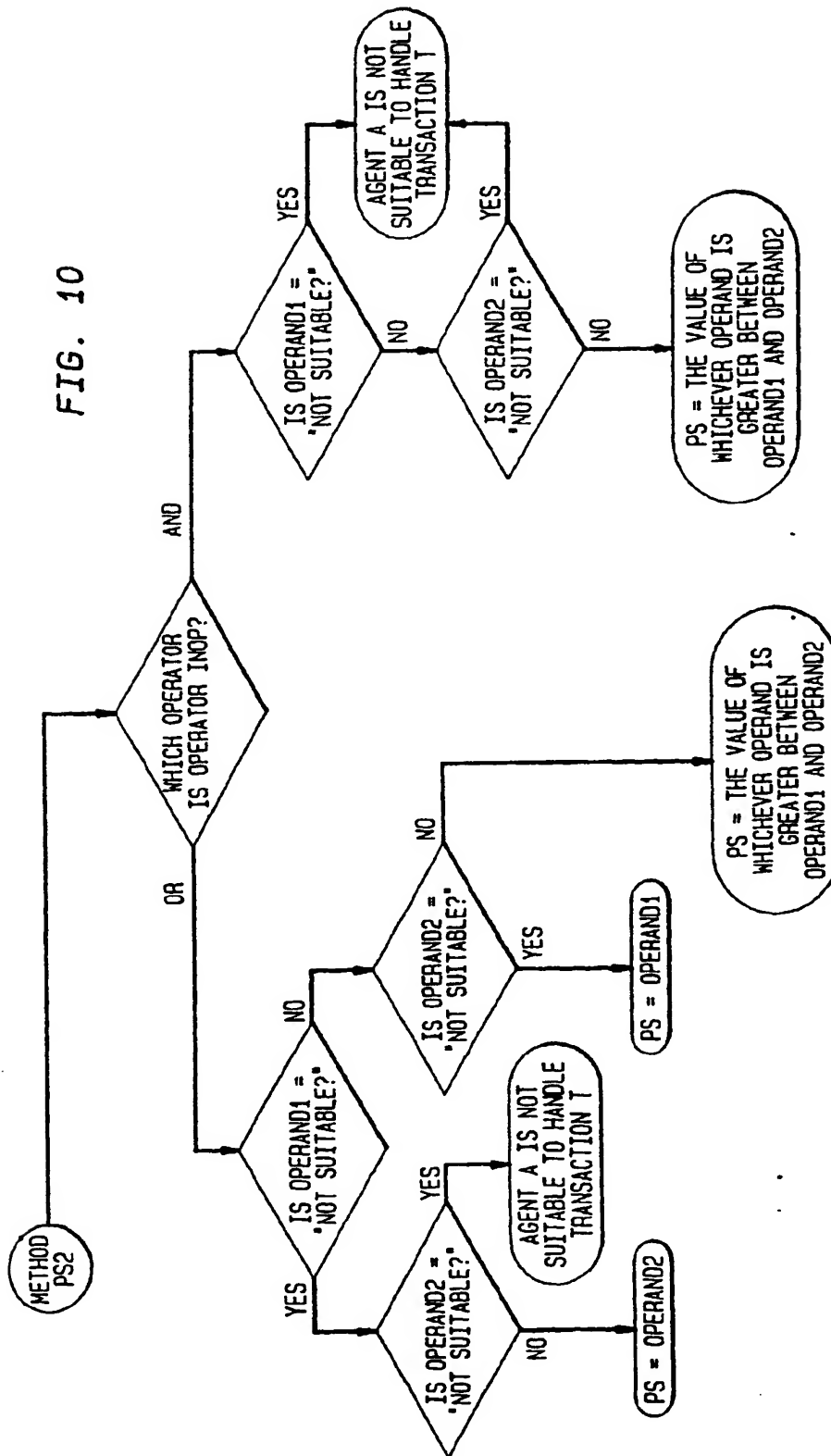
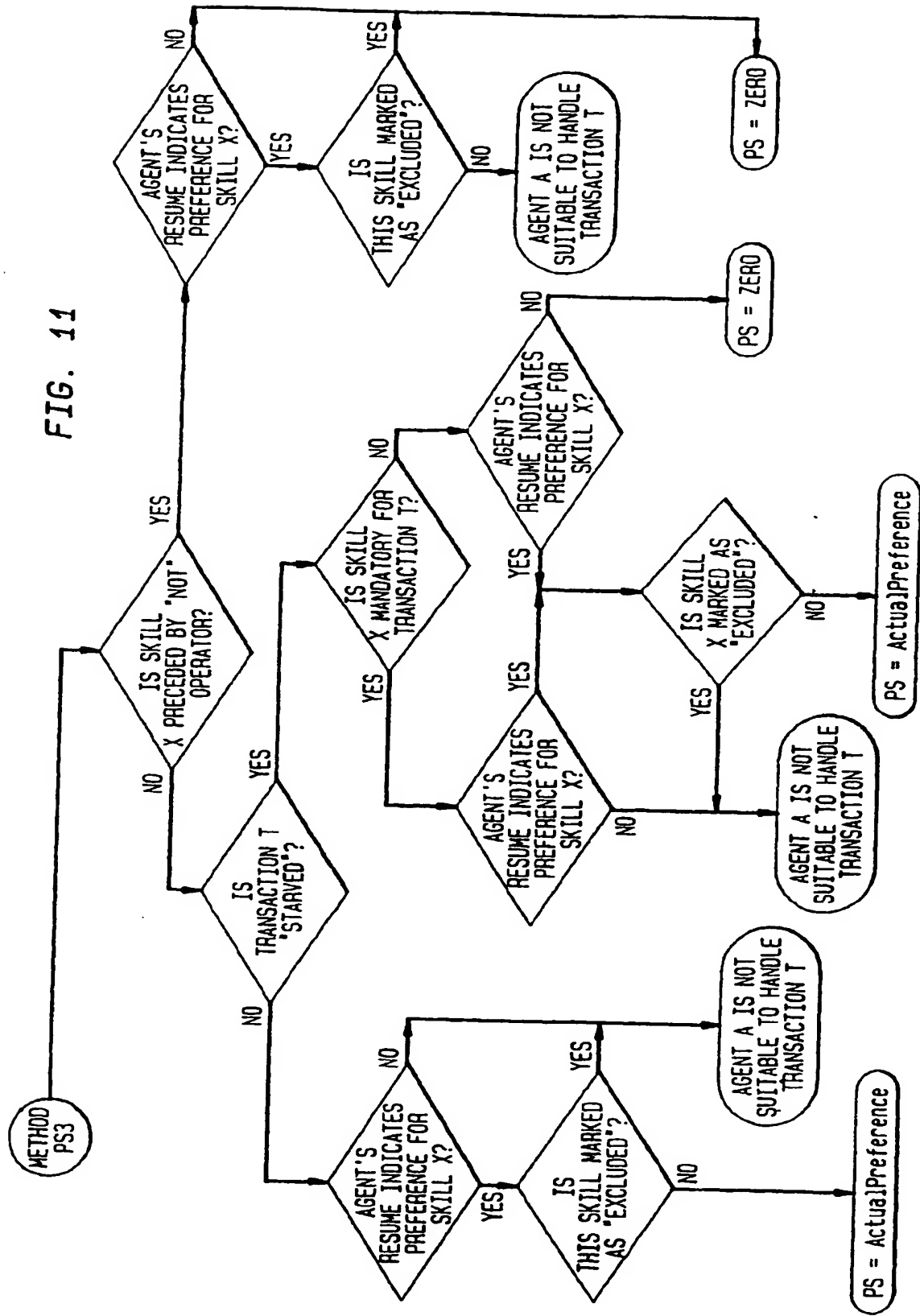


FIG. 11



This Page Blank (uspto)